WEIXING DIANZIJISUANJI JIAOCHENG

微型电子计算机教程

翟才忠 编



APPLE Ⅱ 微型计算机
实习操作指导

广东科技出版社

Retro Workshop
果粉工作室

微型电子计算机教程

APPLE Ⅱ微型计算机
实习操作指导

翟才忠　王晓阳　编

广东科技出版社

内 容 简 介

本书是《微型电子计算机教程》中指导APPLE Ⅱ微型计算机实习操作的学习用书。主要内容有：APPLE Ⅱ PLUS系统的安装和操作方法；监控命令、小型汇编及监控子程序的调用；浮点BASIC语言的程序实例；DOS3.3磁盘操作系统的功能；文件管理方法及应用操作实例。

本书收集了丰富的程序范例，实用性强，可供具有一定微型计算机基础知识的读者或APPLE Ⅱ微型计算机的用户作为实习操作的指南，亦可作为中专、中技以及非电子类专业的大专院校进行微型计算机教学的参考教材。

微型电子计算机教程

APPLE Ⅱ 微型计算机

实 习 操 作 指 导

翟才忠　王晓阳 编

·

广 东 科 技 出 版 社 出 版

广 东 省 新 华 书 店 发 行

广 东 新 华 印 刷 厂 印 刷

787×1092毫米 32开本　10.5印张　200,000字

1985年7月第1版　1985年7月第1次印刷

印数 1—40,000册

统一书号 15182·119　定价 1.60元

出 版 说 明

《微型电子计算机教程》（以下简称教程）是根据目前中专、中技和职大的教学要求编写的。在编写过程中注意按教学和培训的需要来安排内容，力求概念简明易懂，突出使用和操作技巧，便于学习者自行复习，同时注意与实践相结合。《教程》中收入的许多程序实例都能在实践中直接运用，在学习中起举一反三的作用。

本《教程》首批安排出版的选题共有六种，以目前使用范围较广的TRS—80、APPLE Ⅱ（苹果2型）、R1等机型为教学机，分别介绍各型微型机的BASIC语言的编程知识和上机操作方法、系统结构原理和用汇编语言编程方法等内容。这六种选题是：（1）《微型计算机基础》；（2）《TRS—80微型计算机BASIC Ⅱ语言与磁盘操作系统》；（3）《APPLE Ⅱ微型计算机BASIC语言与磁盘操作系统》；（4）《APPLE Ⅱ微型计算机系统结构》；（5）《APPLE Ⅱ微型计算机实习操作指导》；（6）《R1微型计算机读本》。以后根据教学的需要将补充若干选题。其中，《微型计算机基础》是公共基础课教材，其余各书可由各教学单位根据所采用的教学机型号，结合课程设置和教学要求选用。

参加这套《教程》编写工作的作者是大学的教授、讲师及中专老师，他们曾为几十期培训班讲课，积累了微机教学的经验。《教程》中的六个选题就是在他们讲课的基础上经过认真修改补充后写成的。

本《教程》可供中等专业学校和中级技工学校、职工大学和业余大学开设微型计算机课时作参考教材使用，亦可供各企事业单位和大专院校开办微型机短期培训班时用作教材，还可供广大读者用作自学读本。

前　　言

APPLE Ⅱ微型计算机是目前国内外流行的一种微型计算机。为了推广普及微型计算机的应用，我们编写了这本书，供具有微型计算机基础知识的读者和 APPLE Ⅱ PLUS 微型机（或与APPLE Ⅱ　PLUS 兼容的微型计算机）的用户作为实习操作的指导书，亦可供中专、中技以及非电子类专业的大专院校用作微型计算机教学的参考教材。

本书的特点是能够结合操作范例，详细介绍各种操作的具体步骤。读者只要按步骤操作，即可重现程序的执行过程。书的主要内容有：APPLE Ⅱ　PLUS 系统及安装、操作方法；监控命令、小型汇编及监控子程序的调用；浮点 BASIC（即APPLE　SOFT）程序设计语言和磁盘操作系统（DOS 3.3）命令；汉字卡（字库）及打印机的使用方法等。多数章后附有相应的编程范例和习题，供读者上机练习和思考，以加深理解和尽快用好APPLE Ⅱ　PLUS 微型计算机。

本书第一至第六章、附录一以及附录四至附录十二由翟才忠编写，第七至第十二章以及附录二由王晓阳编写，附录三由高倩编写。全书由高倩校核，李人翼审定，借此一并致谢。

因编者学识水平所限，书中一定有不少缺点、错误，希望读者批评指正。

编　者

一九八四年九月于广州

Retro Workshop
果粉工作室

目　　录

第一章　基本系统的构成及安装……………………………（1）
　第一节　基本系统的构成 ……………………………（1）
　第二节　系统的安装连接 ……………………………（5）
第二章　APPLE Ⅱ微型计算机的操作 ………………（8）
　第一节　开机与关机 …………………………………（8）
　第二节　屏幕显示 ……………………………………（9）
　第三节　键盘……………………………………………（10）
　第四节　卡式磁带机的使用……………………………（14）
　第五节　软磁盘驱动器的使用 ………………………（16）
第三章　机器语言监控命令的使用 ……………………（20）
　　一、检查存贮器内容的命令 …………………………（20）
　　二、改变内存单元内容的命令 ………………………（22）
　　三、移动一段内存单元内容的命令 …………………（22）
　　四、比较两段内存单元内容的命令 …………………（23）
　　五、将一段内存地址的内容存入磁带的命令 ………（24）
　　六、从磁带取出一段内容存入内存的命令 …………（24）
　　七、运行机器语言程序的命令 ………………………（25）
　　八、列出机器语言及汇编语言程序清单的命令 ……（25）
　　九、检查和修改CPU寄存器内容的命令……………（25）
　　十、改变显示方式的命令 ……………………………（26）
　十一、键盘命令 …………………………………………（26）
　十二、打印命令 …………………………………………（26）

十三、用户自己定义命令 ……………………………………（27）
十四、十六进制的加减运算 …………………………………（27）
第四章　小型汇编及调用监控子程序 …………………（28）
第一节　小型汇编 ………………………………………（28）
第二节　小型汇编程序指令的格式 ……………………（31）
第三节　监控子程序的调用……………………………（33）
第四节　监控程序中一些有用的子程序 ………………（34）
第五节　监控程序的特殊地址…………………………（40）
第六节　小结……………………………………………（41）
第五章　FP　BASIC 状态 ………………………………（46）
一、FP　BASIC 的基本符号及其用法说明……………（46）
二、FP　BASIC 中数的表示法…………………………（48）
三、变量 …………………………………………………（49）
四、运算表达式 …………………………………………（51）
第六章　FP　BASIC程序命令和基本程序语句…………（52）
第一节　立即命令 ………………………………………（52）
第二节　基本程序语句 …………………………………（53）
一、打印输出语句 ………………………………………（53）
二、赋值语句 ……………………………………………（53）
三、键盘输入语句 ………………………………………（54）
四、键盘单字输入语句 …………………………………（54）
五、成批赋值语句 ………………………………………（56）
六、无条件转移语句 ……………………………………（60）
七、循环语句 ……………………………………………（61）
八、注释语句 ……………………………………………（67）
九、暂停语句 ……………………………………………（67）
十、结束语句 ……………………………………………（68）
十一、变量清除语句 ……………………………………（68）

十二、清屏并归位语句……………………………………（69）
十三、置正文显示方式语句…………………………………（69）
十四、置反转显示方式语句…………………………………（70）
十五、置闪烁显示方式语句…………………………………（70）
十六、置正常显示方式语句…………………………………（70）
十七、程序跟踪与不跟踪语句………………………………（70）
十八、输出显示速度的控制语句……………………………（71）
综合例题……………………………………………………（72）
习题与思考题………………………………………………（76）
第七章　FP　BASIC 的其它语句……………………………（78）
一、水平移动光标语句………………………………………（78）
二、垂直移动光标语句………………………………………（78）
三、本行移动光标语句………………………………………（78）
四、相对定位打印语句………………………………………（79）
五、取得目前光标的水平位置语句…………………………（79）
六、调用机器语言子程序的语句……………………………（81）
七、条件等待语句……………………………………………（81）
八、内存取语句………………………………………………（81）
九、传数及执行机器语言子程序语句………………………（81）
十、存贮空间定限语句………………………………………（82）
十一、查询可用的内存量语句………………………………（82）
十二、数组定义语句…………………………………………（82）
十三、数字（符号）变字符串语句…………………………（86）
十四、字符串变成数字（符号）语句………………………（86）
十五、ASCII码变成字符语句………………………………（87）
十六、字符变成ASCII码语句………………………………（87）
十七、求字符串长度语句……………………………………（88）
十八、左取字符串语句………………………………………（88）
十九、右取字符串语句………………………………………（88）

二　十、中间取字符串语句 ……………………………（88）
二十一、条件转移语句 ……………………………（91）
二十二、调用子程序语句 ……………………………（99）
二十三、返回主程序语句 ……………………………（99）
二十四、开关语句 ……………………………（108）
二十五、自定义函数语句 ……………………………（113）
习题与思考题 ……………………………（117）
第八章　FP　BASIC的图形、发音功能及其程序设计
方法 ……………………………（121）
第一节　低分辨率图形 ……………………………（121）
一、设置低分辨率图形语句 ……………………………（121）
二、选择低分辨颜色语句 ……………………………（121）
三、定位画点语句 ……………………………（122）
四、绘水平线语句 ……………………………（122）
五、绘垂直线语句 ……………………………（122）
六、定点查色值语句 ……………………………（122）
第二节　高分辨率图形 ……………………………（130）
一、设置高分辨率图文混合型语句 ……………………………（130）
二、设置全屏高分辨图形语句 ……………………………（130）
三、设定绘图颜色语句 ……………………………（130）
四、定位画点语句 ……………………………（131）
五、绘线段语句 ……………………………（131）
第三节　发音功能语句 ……………………………（133）
第四节　游戏控制程序语句 ……………………………（134）
第五节　编程简例 ……………………………（134）
第六节　绘图技巧——图形表 ……………………………（138）
第九章　DOS3.3（磁盘操作系统）使用指导 ……………………………（148）
第一节　基本操作 ……………………………（149）

第二节　DOS系统的基本命令 ……………………（154）
第三节　机器语言的DOS命令 ……………………（160）
第四节　DOS 命令在程序中的使用 ………………（163）
第五节　文本文件的使用 …………………………（165）
第六节　DOS 系统盘中的应用程序 ………………（181）
实习与思考题 ………………………………………（184）
第十章　打印机的使用方法 …………………………（189）
一、打印机简介 ……………………………………（189）
二、EPSON 打印机及其接口板的使用方法 ………（190）
三、GRAPPLER +接口板的使用方法 ……………（195）
第十一章　汉字卡的使用方法 ………………………（200）
一、汉字卡简介 ……………………………………（200）
二、汉字卡的调用 …………………………………（200）
三、仓颉字母汉字组码输入简则 …………………（201）
四、软件功能 ………………………………………（203）
五、子程序的调用 …………………………………（203）
六、调用软开关的POKE 命令 ……………………（204）
七、使用汉字卡时内存的分配 ……………………（204）
八、使用打印机输出中文 …………………………（206）
九、超级汉卡 ………………………………………（210）
第十二章　深入一步的程序设计技巧……………………（212）
第一节　显示的技巧……………………………………（212）
第二节　APPLE Ⅱ 高速图形及屏幕绘图的新方法
……………………………………………………………（217）
第三节　编制音乐程序的技巧 ………………………（247）
第四节　提高磁盘效率的技巧 ………………………（259）
附录一　FP BASIC 语言(APPLESOFT BASIC)
摘要 ………………………………………（268）

附录二　FP BASIC 语言的错误信息……………………（286）
附录三　DOS 错误信息 ……………………………………（289）
附录四　6502 的指令系统 …………………………………（292）
附录五　APPLE Ⅱ机内存空间的分配 ……………………（310）
附录六　正文显示屏与显示 RAM 的关系图 ………（318）
附录七　低分辨率图形屏幕与显示RAM的关系图
……………………………………………………………（319）
附录八　高分辨率图形屏幕与显示RAM的关系图
……………………………………………………………（320）
附录九　ASCII 屏幕字符表 ………………………………（321）
附录十　十六进制数与十进制数转换表 ………………（322）
附录十一　十进制数与 BCD 转换表 ……………………（323）
附录十二　相对转移位移量查算表 ………………………（325）

Retro Workshop 果粉工作室

第一章　基本系统的构成及安装

APPLE Ⅱ微型计算机是一种功能较强，结构灵活，配套硬件、软件丰富，支持开发的硬件、软件实力雄厚，技术资料齐全，使用简单，价格低廉，无特殊环境要求的中低档微型电子计算机。

第一节　基本系统的构成

APPLE Ⅱ微型计算机的基本系统包括三项设备：主机、与主机连接的键盘及显示屏幕。下面对这些必须的设备以及一些可供选择的外围设备作概括的介绍。

1.主机

在计算机系统中，用户是通过主机控制所有外围设备的。APPLE Ⅱ主机的内部结构如图1-1所示。将主机的顶盖打开，就可以看到一块绿色的大印刷电路板（称为主板），它占据机箱底的大部分面积。主机上装有约80块集成电路片（IC）和一些其它元件。在主板上有8个标准扩充插座（25×2脚双列直插式），通过这些插座，APPLE Ⅱ微型机能与50多种接口板和外围设备连接，以扩充系统的功能。在8个标准扩充插座的下方，有一块大一点的40脚IC，这就是微处理器(CPU)6502。微处理器下方的6个24脚的IC插座，最多可插入12K字节容量的只读存贮器ROM或EPROM。监控程序以及FP BASIC（浮点 BASIC）解释程序就存放在

ROM中。ROM的下方装有每排8个、共三排的随机存贮器RAM插座（见虚线框内）。可以插入容量为16K×1位的动态RAM芯片，但插入的每排8个芯片的型号必须相同，以组成48K字节容量的RAM。键盘送入的信息就存放在RAM中。

主板上的其他元件还有各种功能，如控制信息从微型计算机的一部分传送到另一部分，从外部采集数据，送信息到屏幕去显示或发出声音等。

主机由稳压电源供电。

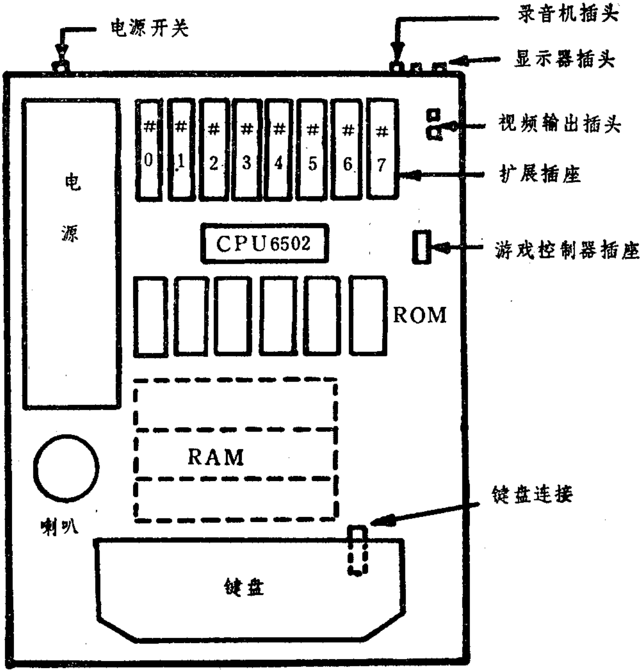


图 1-1 APPLE Ⅱ微型计算机内部结构方位图

2.键盘

键盘是APPLE Ⅱ微型机最基本的输入设备。APPLE Ⅱ键盘有标准ASCII打字键51个（详见第二章第三节）。

3.显示器

显示器是APPLE Ⅱ机最基本的输出设备。显示器不但显示由键盘输入的指令信息，而且忠实地表达APPLE Ⅱ机对指令的反应。APPLE Ⅱ机的显示器目前多数采用12英寸绿色显示器（MONITOR）。当然，也可以使用一般的彩色电视机或黑白电视机，不过黑白电视机不能照原色显示APPLE Ⅱ主机提供的彩色图形。

4.内存贮器（内存）

微型计算机的内存贮器由许多小单元——字节（byte）组成。存贮器的每一个字节可存贮一个字母或8位二进制数。APPLE Ⅱ存贮器的容量，可由8K增加到64K字节（1K=1024个字节）。存贮容量往往决定APPLE Ⅱ机所能从事的工作性质。

APPLE Ⅱ的内存具有两种形式。一种称为ROM（Read Only Memory）——只读存贮器。即使关闭机器电源，它的内容也不会改变。ROM保存有APPLE Ⅱ的系统监控程序（2K）和FP BASIC解释程序（10K）。另一种称为RAM（Random Access Memory）——随机存贮器，它的内容可以被我们修改，关闭电源，信息会消失。RAM是供用户存贮所输入的程序或数据的。

5.卡式磁带录音机（外存）

卡式磁带录音机可用于贮存程序及数据，并可与RAM作各种数据的交换和存取。因此我们可以用普通录音机作为APPLE Ⅱ的外存贮器来存放许多数据及程序。卡式磁带录

音机的耳机（EAR）及话筒（MIC）插头，应分别插入微型计算机右后部的两个录音机插座中。

6.软磁盘驱动器（外存）

用磁盘驱动器（DISK DRIVE）作为外存贮器远比用磁带录音机好，因为磁盘驱动器处理数据的速度快、可靠性高、存贮量大，且能自动存贮。通常APPLE Ⅱ机的磁盘驱动器只用单面双密度的软磁盘，每个盘可提供140KB的容量。磁盘驱动器通过线排接到驱动器接口板，再接到主机的#6插座（或#5插座上）。每块驱动器接口板可连接两台驱动器。

7.温式硬磁盘机（外存）

通过CORVUS硬磁盘系统接口板，APPLE Ⅱ机可接配容量更大的温式硬磁盘机。硬磁盘可有5MB、10MB、20MB等三种容量，它能够配用FP BASIC、APPLE PASCAL和APPLE FORTRAN等高级语言。另外，经过“多工器”还可以接成一套最多可达64部APPLE Ⅱ的星形网络。

8.打印机（输出设备）

与APPLE Ⅱ机配接的打印机，通常是并行打印机。打印机的型号有多种类型。如EPSON打印机，可打印中、英文及图形等，可通过EPSON并行打印接口板连接至#1扩充插座。又如μ-80打印机（只能打英文），也可通过EPSON并行打印机接口板连接至#1插座。还有CENTRONICS 779打印机，可通过CENTRONICS打印接口板插入#1插座。

APPLE Ⅱ机也可以通过串行打印接口板，连接串行打印机。

9.游戏控制器（内部设备）

游戏控制器作为APPLE Ⅱ的一种内部设备，它用于游

戏控制，可作为一种娱乐工具。通常它并不与APPLE Ⅱ机连接，使用时要把它接到主板上的游戏控制器插座上。

10.彩色绘图板（输入设备）

彩色绘图板是由APPLE公司推出的，它将APPLE Ⅱ机的绘图能力用特殊的方法表现出来——可以用手直接操作。用它的"笔"可随意在绘图板上画出各式各样的彩色图形，并显示在屏幕上；也可利用它的软件包配置固定功能，按照使用者的要求画出点、线或方格。图形可以占用整个屏幕或只占其一部分，也可以移到使用者希望的任何位置。图形可以放大、缩小或作分色处理。而在处理过程中，可以随时将图形贮存在磁盘上，供以后使用。至于计算图形的面积或距离的测量，可以利用绘图板的特殊功能来实现。绘图板接口可插入#1至#7的任一插座中。

第二节　系统的安装连接

APPLE Ⅱ微型机系统的安装很简单，主要是外围设备与主机及扩充板与主机的连接，各种连接点如图1－1所示。

1．显示器的连接

将与显示器连接的电缆的另一端插头，插入主板右上角标有VIDEO的插孔中。显示器通常直接接220伏市电，但有些则需通过一个电源转换器才能与220伏电源相接。使用时务必注意。

2．卡式磁带录音机的连接

将录音机的EARPHONE（耳机）输出插头，插入主板右上角的"IN"插口；将录音机的MICROPHONE输入线，连到主板右上角的"OUT"插口。录音机可直接接220伏市电，

或经稳压器再接220伏电源；建议不要用干电池给录音机供电（因为干电池损耗较快，不能恒压，以致不能保证录音机马达以恒速转动，这样会影响信息存取的可靠性）。

3．软磁盘驱动器的连接

通过连接线排接到驱动器接口板上的22脚双列针式插头（注意线排塑料接头的三角符号向外侧）上。每块驱动器接口板有两个插头，可分别接两台驱动器（DRIVE 1和DRIVE 2）。若配接两台驱动器，则一块驱动器接口板插入#6插座，而驱动器的自然编号为1和2；若配接四台驱动器，则第二块驱动器接口板插入#5插座，而这台驱动器的自然编号为1号和2号，以此类推。APPLE Ⅱ机最多可配接6块驱动器接口板（即12个软磁盘驱动器）。软磁盘驱动器的电源由主机供给。

4．并行打印机的连接

以EPSON（FX-80，MX-80，RX-80，CP-80，DP-80，RP-80或STAR等）为例，把打印机连接线排接入打印接口板，而将打印接口板插入主板#1插座即可。

5．其它扩充板的连接

若要扩充其它外围接口板，则将接口板对应插入下列外围设备插座：

#0插座：16KRAM（语言板），不能插入其它任何板；

#1插座：打印机；

#2插座：EPROM写入器或其它板；

#3插座：中文字库板（汉字卡）或其它板；

#4插座：Z-80板（或称SOFT CARD），或声音识别器板；

#5插座：绘图板，80字符板或其它板；

#6插座：磁盘驱动器接口板；

#7插座：PAL板（连接国产PAL制TV）。

还有其它多种接口板，它们均可以插入 #1 至 #7 插座，而#0插座只能插16K RAM板或留空（因为这个接口是供键盘与显示器专用的）。同时，应该记住各外围设备对应的标准插座号，因在使用（操作）中经常要用到。

将各种外围设备连接好后，再将所有连接线从主机后面齿状缝中引出，整理摆放好，然后盖上主机顶盖，把主机电源开关置于“OFF”（关）的位置。这样，整个系统就安装好了。经查对连接无误后，微型计算机便可通电使用。

第二章　APPLE Ⅱ微型计算机的操作

第一节　开机与关机

开机前，先检查系统的安装连接是否正确，确认无误后才可开机。

一、开机顺序

开机必须遵守下述顺序：

（1）打开显示器开关；

（2）打开打印机开关（若不使用打印机，则不需打开）；

（3）把系统主磁盘（DOS3.3）插入#6插座的第1号驱动器内，并封盖（若不使用磁盘则不需插入）；

（4）打开主机电源开关。

二、关机顺序

关机必须遵守下述顺序：

（1）取出驱动器中的磁盘(或把驱动器封盖门打开)；

（2）关闭主机电源开关；

（3）关闭打印机；

（4）关闭显示器。

此外，开机与关机（尤其是开主机与关主机）的动作切不可太频繁，否则，将会损害机器。由关机到再开机的时间间隔，最好在两分钟以上。

当打开主机电源开关时，键盘左下角的“POWER”灯亮。若已接上磁盘驱动器，则驱动器上的“正在使用”灯亮，表示驱动器在工作。当驱动器工作完后或当按下CTRL-RE-

SET键时，计算机就会发出“哔”的响声，屏幕上显示一个“]”符号（这是FP BASIC 的提示号），表示计算机已准备就绪，可以使用FP BASIC语言和磁盘操作系统(DOS)来命令计算机工作了。

若系统没有安磁盘驱动器，一旦开主机电源，计算机立即进入FP BASIC(浮点 BASIC)状态，并发出“哔”的响声。

如果听不到“哔”的响声，应先将主机关闭，然后再重新开机，若仍听不到任何声音，则再关主机，查看电源连接是否正确，接触是否良好，市电电压是否在220±10%伏的范围内。如果这些方面都没有问题，则需重新查看所有外围设备的连接是否正确，接头是否接触良好等，然后再重新开机。如果上述步骤都不能使APPLE Ⅱ机有所动作，此时必须立刻关机，请APPLE Ⅱ机维修站的专家协助检查。不要擅自乱动机器内部部件，尤其千万不能带电（未关电源时）乱动机器内部部件（如拔插IC片或外围设备接口板等），因为这将会导致不可预料的损坏。

第二节 屏幕显示

一、光标

当开机后听到“哔”的响声，就表示机器正常，这时在屏上将显示出一些特定的信息。

由于APPLE Ⅱ机的类型不同，屏上的这些特定信息也有不同，但最后在屏上闪烁的小方块（称为光标）是相同的，它所在的位置就是所需要键入的下一个字的位置，在光标之前有一个FP BASIC的提示号“]”。

有两种情况是看不到光标的，下面介绍看不到光标时的处理办法。

（1）若在APPLE Ⅱ主机的#0插座中插入了16K RAM板（语言板），并且与一台或多台磁盘驱动器连接，开机后就会看不到光标，只是听到磁盘机转动的声音，看到磁盘机的指示灯发亮。在这种情况下，需按RESET键或CTRL-RESET键，使磁盘机停止转动才能显出光标。

（2）若APPLE Ⅱ机内接入80字符板（假定插入 #5插座），此时必须做以下操作才能看到屏上的光标：

1）按住CTRL键不放，再按下B键（即CTRL-B），再同时放开；

2）键入PR #5，按RETURN键。

这时光标将会出现，虽然它可能不闪动，但表示一切都已正常。

二、系统的几种状态提示符号

在光标左边的符号就是某种系统状态或语言状态的提示符号，它们分别是“ * ”，“] ”，“>”，“OK”或“ ! ”。其中：

“] ” 表示FP BASIC（浮点BASIC）状态

“ * ” 表示监控状态（监控命令状态）

“>” 表示INT BASIC（整数BASIC）状态

“ ! ” 表示小型汇编状态

“OK” 表示MBASIC或GBASIC状态

第三节 键　盘

在APPLE Ⅱ微型计算机上运行的大多数程序及命令，是由键盘输入，也就是说键盘是人机联系的一种输入设备。

从外形看来，APPLE Ⅱ的键盘与标准英文打字机的键盘相似，但较一般打字机多了七个键（见图2-1）：ESC键、CTRL键、RESET键、←键、→键、RETURN键和



图2-1 APPLE Ⅱ 微型计算机键盘

REPT键。

下面分别介绍APPLE Ⅱ键盘各键的功能。

1．26个英文字母键

26个英文字母键可以有大小写控制。当POWER灯键被按弹起时，可分别输入大写英文字母；当POWER灯键被按下时，可分别输入小写英文字母。

2．10个数字键

0至9共10个数字键可分别输入10个数字。为了区分数字“0”与字母“O”，数字“0”在屏幕上显示为“Ø”。请读者注意，本书中数字“0”的印刷符号仍表示为“0”。

3．符号键

符号键能产生各种运算符号、标点符号。

4．RESET键

这是复位键，它是一个特殊的键。按下RESET键时，微型计算机将停止处理正在处理的事情，并把系统控制权交回给键盘，这时系统回复到前一种语言状态。

有时错按RESET键会产生一些问题，尤其是在使用磁盘驱动器时，不应随意地按RESET键。为了避免按RETURN键时错按了RESET键（因为这两键较近），有许多微型计算机的RESET键要与CTRL键同时使用（可参看以下CTRL键）。

5．RETURN键

这是回车换行键，用以结束已输入的一行命令或字符串。当在键盘上键入字符时，所键入的字符显示在屏幕上，这时微型计算机也将输入的信息存于内存里，但未进行编译工作，只有按下RETURN键，编译工作才开始。

按下RETURN键，计算机就知道用户已输入完一行程

序，并自动将光标以后的字消除，然后去辨认用户所键入的这一行程序。如果它能认识用户所键入的程序中的每个字符，它将会作出正确的反应，否则，计算机将发出“哔”的一声，并在屏幕上显示出如下错误信息：

? SYNTAX ERROR

这时，必须重新进行输入操作。

在本书以下的叙述中，每输入一个程序行或一条命令，如果没有特别说明，都要按RETURN键。它的缩写是↲。

6. SHIFT键

SHIFT键与某键配合同时使用，是指启用某键的上部功能。“同时使用”通常以一段短连线“-”来表示，如 SHIFT-8 得到的是“#”，SHIFT-P则显示“@”。但有些字符键并不显示其上部符号，如SHIFT-G，G键的上部是 BELL 字，但并不显示“BELL”，而是显示“G”。

7. CTRL键

CTRL是CONTROL（控制）的缩写。象 SHIFT 键一样，CTRL键要与其他键同时配合使用。操作上也是先按住CTRL键，然后再按其他键，直到放开后者，才放开CTRL键。书写上也是以 CTRL-来表示。例如，CTRL-B 表示按CTRL键及B键。

CTRL键也象SHIFT键一样，由于它的配搭使用，而增加了一些键的功能。例如：CTRL-B可使微型计算机由监控状态转入FP BASIC语言状态；CTRL-G可使微型计算机响“哔”的声音；CTRL-X 会告诉计算机把用户刚打入的一行作废（光标移到该行的最左边，让用户重新键入）；CTRL-C可中断正在执行的程序或正在执行的列表 LIST 命令。

8. ESC键

ESC 键是 ESCAPE 的缩写。该键与某些键配合主要用于键盘编辑。

ESC 键与 CTRL 键及 SHIFT 键不同的地方在于：当使用 ESC 键时不必一直按住 ESC 键，而是按下后可以立即放开，再按需要配对的键。

例如：ESC-@，首先按下 ESC 键，然后放开它，再按下 SHIFT-P（SHIFT-P 为@）。此时屏幕上的一切显示信息都清除了，只留下光标在左上角闪动（在计算机的术语中，屏幕的左上角叫做 HOME）。

若想不再使用 ESC 键，则再按一次 ESC 键，即可解除。

与 ESC 键配合的键还有许多，如控制光标上、下、左、右移动等屏幕编辑的功能键，详见本书键盘编辑的章节内容。

9. ←键与→键

左箭头键与右箭头键可用于更正任何打错的信息或更正已经打入的信息。

每按一次←键，光标就会向左倒退一格，而光标倒退时经过的字符就不再存于用户内存中（当然仍留在显示 RAM中）。

每按一次→键，光标就向右移一格，而光标往右移动时所经过的字符就如同从键盘上打入一样，被存入了用户内存区。

10. REPT 键

REPT 就是 REPEAT（重复）的缩写。如果同时按住 REPT 键及任何一个键，这两个键就会连续地重复显示在屏上，直到用户放开其中一个键或同时放开这两个键为止。用于快速编辑时，←键或→键也可以与 REPT 共用。

第四节　卡式磁带机的使用

1. 卡式磁带机与主机的连接

请参见第一章第一节（卡式磁带机）。

2．磁带的保养与使用

由于磁带很容易受损坏，并且不容易修复，所以在使用和保养时必须注意如下几点：

（1）防油污、防潮，不要碰磁带表面（尤其不能用带有油脂、尘粒、潮气的物体碰触磁带表面）；

（2）防弯折，不要折叠磁带；

（3）防高温，不可将磁带存放在高温（高于36℃）和太阳直射的地方；

（4）防磁，不可将磁带存放在有交变磁场的地方。

3．卡式磁带录音机音量的调整

卡式磁带录音机的存取与音量的关系极大。音量过大或过小，都将使微型计算机与卡式磁带录音机的存取无法进行或存取不可靠。

对于不同的微型计算机或不同的卡式磁带录音机，所选取的音量是不相同的。调整音量的方法是反复试验和修正，步骤大致如下：

首先将音量调至最低或较低，然后试将程序输入给计算机。如果不能成功，则将音量稍稍提高，再试做存取工作；如果仍然不成功，则继续将音量提高，再进行存取；如此反复调整，直到成功地将程序可靠存入或取出为止。此时，请记住这套机配这台卡式录音机的音量位置。

若存取操作有时成功而有时不成功，则可以用磁头清洁剂或纯酒精清洁录音机的磁头，或更换另一盒磁带。

4．磁带机的存取操作步骤

（1）把从磁带取出的程序存入微型计算机内存的步骤如下：

1）将磁带转回开始的位置；

2）在微型计算机键盘上键入 LOAD 命令；

3）按下微型计算机的 RETURN 键。

4）在卡式录音机上按下 PLAY 键，使磁带开始转动；

当按下 RETURN 键后，光标会消失；经过大约十几秒，就可以知道存取是否成功。

（2）把微型计算机中（内存）的程序存入磁带的步骤如下：

1）准备好一段空白磁带，并记住开始位置；

2）键入 SAVE 命令；

3）按下录音机的 RECORD 及 PLAY 键；

4）按下 RETURN 键。

过一段时间，即可以知道存取是否成功。

若最后听到“哔”的响声，而屏上又没有什么信息显示，则表明存取成功了。

若屏上显示“? SYNTAX ERROR”，则不需再调音量，只要再按以上的步骤重做一次。若还不成功，则可能是磁带或磁头的问题。

若微型计算机没有任何反应（迷路）或显示“ERR”等信息，则必须按 RESET 键，并重新调整录音机的音量，然后再试，直至成功为止。

第五节　软磁盘驱动器的使用

1．软磁盘驱动器与主机的连接

详见第一章第二节。

2．软磁盘的保养及使用

由于软磁盘片较磁带更加脆弱，它表面上有一层十分柔

软而又质薄的磁氧化物，所以在手拿、使用和存放磁盘时必须特别小心，否则磁盘是很容易被损坏的。为了延长软磁盘的使用寿命，要注意以下十个方面。

（ 1 ）不要触摸盘面上的磁氧化物，特别不要触摸封套开口处的磁头槽。因为手指上的油质可能使磁盘上的数据永久不能恢复。

（ 2 ）不要把磁盘平放在脏的、有灰尘的地方。因为磁头和磁盘之间即使有一小粒灰尘，也将会在磁氧化物涂层上划出痕迹。因此，不使用的磁盘应立即将它放回封袋内。

（ 3 ）操作员在上机工作时不要吸烟。因为空气中的烟雾和尘粒会聚集在磁盘上，而使盘面受损。

（ 4 ）不要用圆珠笔等尖硬的东西直接在磁盘标签处及封袋上写字，只能采用较柔软的笔来写，而且书写时压力要很小，以免损坏磁盘表面。

（ 5 ）不要擦去已写在磁盘上的标记，因为无论擦时多么小心，橡皮和纸的微粒会弄脏磁盘。

（ 6 ）往驱动器或封袋里装磁盘时不要用力过猛。如果有阻塞，要把磁盘拿出来再装一次。

（ 7 ）不要弯曲磁盘。因为磁盘受弯会导致磁氧化物层脱落，致使磁盘片不能再用。

（ 8 ）不要把磁盘片存放在温度超过38℃或有漏磁场的地方。

（ 9 ）磁盘片保存时不要平放或叠放，要象放唱片一样垂直存放在防尘盒内。

（10）当磁盘片在驱动器内时，不要开、关主机电源（即磁盘系统的电源）。因为开、关电源会使驱动器马达产生漏磁场，从而可能破坏磁盘上的数据。

遵守上述规则，就可大大延长驱动器及磁盘的工作寿命。

3．磁盘的插入方法

打开磁盘机的门盖后，可用拇指及食指拿住磁盘封套袋的贴标记的一侧（拇指在上方），然后轻轻地将磁盘片滑入磁盘机内。一般来说，这个过程都应顺利完成而不会遇到任何阻塞。如果碰到阻塞，可以将磁盘拿出来，重新放入。需要注意的是：在放入磁盘时，要尽量保持磁盘片的水平。当磁盘片进入磁盘驱动器后，就可以将磁盘驱动器的门盖轻轻地关上，这个过程亦应该顺利完成。但如果无法轻易地关门盖，那可能是磁盘片还没有被完全送入驱动器里，这时需将磁盘片再轻轻往里推进，门盖即可关上。如果未能把磁盘片正确插入到驱动器内而用力将门关上，那将会毁坏磁盘片。有时，可以待磁盘驱动器转动后再关门盖，这样做会容易些，因为磁盘驱动器转动后会使磁盘片放在磁盘驱动器的中央位置。

4．磁盘操作系统的使用

在APPLE Ⅱ微型计算机上，目前可分别运动的操作系统有三个：DOS3.3，CP/M2.2和PASCAL。这些就是通常指的系统软件。下面介绍引导DOS3.3及CP/M2.2的方法。

（1）DOS3.3的引导

DOS3.3是APPLE Ⅱ的主操作系统。使用DOS3.3的条件最好是具备16K RAM板(语言板)，因为这能使主内存扩大到64K，从而能把系统主盘中INT BASIC语言的解释程序也引导入机内供使用。

把DOS3.3引导进入内存（英文称为BOOT DOS），有如下两种方法：

1）开机自动引导。在关机状态下，把DOS系统主盘插

入#1驱动器内，并关上门盖，然后打开主机电源，于是，DOS3.3就会自动地被引导进入RAM区，经几秒钟后，引导完毕，在屏上最后的标志是“] ”（这是APPLE Ⅱ引导完DOS3.3后，又回到了FP BASIC状态的提示符号）。

2）PR命令引导。在开机（工作状态下），当想要使用DOS3.3系统（或原有的DOS3.3已消失，要重新恢复）时，可以不关机（以免把你的程序冲掉），在FP BASIC（] ）或INT BASIC（ > ）状态下，只要把DOS3.3系统主磁盘插入#1驱动器内，关上门盖，然后用命令：

PR #6↙（假定驱动卡插入#6插座）或 IN #6↙

便可把DOS引导入APPLE Ⅱ的内存中。

若APPLE Ⅱ上已有16K RAM板（语言板），则在BOOT DOS过程中，INT BASIC的解释程序已引导进入16K RAM内。

在监控状态（ * ）下，可用C600 G ↙来引导DOS。

内存中有了DOS3.3，就可以使用DOS命令了（详见DOS命令一章）。

（2）CP/M2.2的引导

若要使用CP/M操作系统，APPLE Ⅱ也可以满足需要。由于CP/M是芯片为Z-80的磁盘操作系统，因而在具备Z-80板（通常插入#4插座）及16K RAM板的APPLE Ⅱ微型计算机中，CP/M才能运行。把CP/M引导进入内存的方法与引导DOS3.3的方法相同，即开机自动引导和PR命令引导两种方法。

CP/M被引导进入内存后，就可以使用CP/M DOS命令了（详见CP/M磁盘操作系统使用方法一章）。

第三章　机器语言监控命令的使用

在 APPLE Ⅱ PLUS 微型计算机的 ROM 内，有一个自动起动和管理 APPLE Ⅱ 的控制程序，也就是监控系统（MONITOR）。监控程序是用机器语言设计的，它存放在地址＄F800～＄FFFF 共2K 的内存中。监控系统提供了一系列功能强、使用方便的监控命令，在监控状态下，可以编写、输入和执行机器语言程序。监控系统对 BASIC 语言以至其它高级语言来说，是起连接（桥梁）作用的。

当然，只有当机器进入监控状态才能使用监控命令。前面提出过，监控状态的提示号是"＊"。由开机后的自然状态——FP　BASIC（亦称 APPLESOFT 状态）"]"进入"＊"状态的命令为：CALL-151。

当屏上显示出"＊"监控状态提示号时，便可以使用监控命令了。

以下介绍监控命令的功能及其操作范例，供读者练习。

一、检查存贮器内容的命令

1．检查单个内存地址内容的命令

＊E000↙　（查E000地址的内容）
　E000-30　（显示出该地址的内容为＄30）
＊300↙　（查300的内容）
　300-99　（内容为＄99）

2．检查多个内存地址内容的命令

(1) 检查自当前打开的地址的下一个地址开始，到末地

址为止的一段内存地址内容的命令。

格式：＊·<末址>↙

说明：在＊号之后打入一个"·"号，表示"至"，接着打入需检查的末地址。

例如：＊F83F↙

F83F-30　（地址内容为30）

＊·F850↙　（查＄F840至＄F850的内容）。

屏幕显示每行8个地址的内容：

F840-20　28　F8　88　10　F6　60　48

F848-4A　29　03　09　04　85　27　68

F850-29

＊

(2) 检查某一指定内存区间内容的命令

格式：＊<首址>·<末址>↙

例如：＊F800·F810　（查F800至F810区间的内容）

屏幕显示为：

F800-4A　08　20　47　F8　28　A9　0F

F808-90　02　69　E0　85　2E　B1　26

F810-45

＊　↙

(3) 逐行检查内存单元内容的命令

将需要检查的内存单元首址作单个显示之后，每按下一次 RETURN 键便显示一行(8个)内存单元的内容。例如：

＊0800

0800-99

＊

屏幕显示出8个地址内容：

B9　00　08　0A　0A　0B　09

*　（再按一次RETURN）

屏幕又显示8个地址内容：

0308　00…

二、改变内存单元内容的命令

1. 改变一个地址内容的命令

格式：*‹地址›：‹内容›↙

将某数据存入指定的地址。例如：

*302：7E　（将$0302的内容改为7E）

*302　（查看）

0302-7E　（7E已存入地址$0302中）

2. 改变一些连续内存单元内容的命令

格式：*首址：‹内容›␣‹内容›␣‹内容›␣……

按顺序把需送入指定的一系列连续地址的内容（数据或指令码）打入，两个内容之间用空格棒（␣）间开（一次最多可改变85个地址的内容）。例如：

*1200：00　01　02　03　04　05　06　07

（以上8个数据便进入从$1200开始的8个地址中）

*1200，1207　（查看）

1200-00　01　02　03　04　05　06　07

*

三、移动一段内存单元内容的命令

格式：‹目标首址›<‹原首址›·‹原末址›M↙

其中："<"表示移动方向；"·"表示"至"；"M"表示启用MOVE命令。

例如：*1200<F800·F900M

以上操作便把F800～F900的全部内容，抄到地址为

＄1200开始的一段地址中（至1300止）。

读者可以进行验证。

移动一段内存单元内容的命令的特殊应用：

(1) 若“原末址”小于“原首址”，则执行结果只移动（抄写）首址的一个内容。

例如：*1400<F900・F805M

(2) 若“目标首址”是在“原地址范围内”，则可用之创造一种重复值——两个首地址之间的内容当作一个子区。这个子区的内容，将充满整个原首址至原末址的内存区间。

例如：用11　22　33充满300～320内存空间。

```
*0300：11　22　33
*0303<300・320M
*0300-11　22　33　11　22　33　11　22
 0308-33　11　22　33　11　22　33　11
 0318-11　22　33　11　22　33　11　22
 0320-33
*
```

例如：用00充满＄0C00～0C20内存空间（即把该空间清零）。

```
*0C00：00
*0C01<0C00：0C20M
```

以上操作便可把＄0C00～＄0C20段的内存区清零：

*0C00・0C20　（查看）

四、比较两段内存单元内容的命令

格式：(第二段首址)<(第一段首址)・‹第一段末址›V↵其中：V表示启用VERIFY命令。比较结果若一致（相同），则通过，最后显“*”号；若不一致，则显示不一致的地

址及其内容。

例如：＊24D0<300·320V

这时将有不一致的内容及其地址显示于屏幕上。但若把这两段内存地址的内容"复制"后，再予比较，例如：

＊24D0<300·320M

分别将两段内存地址的内容显示于屏幕上：

＊300·320

＊24D0·24F0

这时，比较结果完全一致。

五、将一段内存地址的内容存入磁带的命令

格式：‹首址›·‹末址›W　REC-PLAY

其中：REC-PLAY是录音机的录音键；W表示启用 WRITE 命令。

监控命令能写10秒钟的引导声音（格式化），然后写数据(信息)。平均速度为1350位/秒，存完4K 字节约需要35秒。存完一段内容后，还会自动将"检查总和"（CHECKSUM）值写入，以备读命令，READ用于读出校验。

例如：＊2300·25FF　W　REC-PLAY

执行这一命令，便可把$2300～$25FF的内容存入磁带。

六、从磁带取出一段内容存入内存的命令

格式：‹首址›·‹末址›R　PLAY

其中：PLAY键是录音机的放音键；

"R"表示启用 READ 命令。

例如：＊2300·25FF　R　PLAY

命令执行后，便可把对应磁带段的内容读出，并存入地址为$2300～$25FF的内存中。读出后，也做一个"校验总

和”（CHECKSUM）与WRITE命令的总和相比较。若结果一致，则显示“*”；若结果不一致，则显示错误信息“ERR”，并发出“嘟”的响声。

七、运行机器语言程序的命令

格式：*<地址>G

其中：地址是指要运行的机器语言程序的入口地址（十六进数）；“G”表示启用G0命令。

例如：*3D0G

八、列出机器语言及汇编语言程序清单的命令

格式：*<地址>L

其中：“L”表示启用LIST命令。

用此命令，可从指定的地址开始，显示满屏的指令（20行）。屏左边显示的是机器码，右边显示的是汇编程序指令符。

每个L显示一屏，当要连续显示输出若干屏时，可连续键入若干个L。

例如：*3D0L

九、检查和修改CPU寄存器内容的命令

格式：CTRL-E （即同时按下CTRL及E键）

其中：“E”表示启用EXAMINE命令。

6520的CPU中有5个内部寄存器可供查看和修改。它们分别为：A（累加器）、X（变址寄存器）、Y（变址寄存器）、P（状态寄存器）、S（堆栈指示器），按顺序排列。

例如：*CTRL-E

A=0B X=00 Y=D8 P=7C S=F8

*：B0␣FF␣99␣77␣F4 （修改）

*CTRL-E （查看）

A = B0　X = FF　Y = 99　P = 77　S = F4

*

十、改变显示方式的命令

（1）选反转显示方式（INVERSE）

格式：*I

（2）选正常显示方式（NORMAL）

格式：*N

十一、键盘命令

格式：n　CTRL-K　（即同时按CTRL及K键）

其中：n为插座号（n=1～7）。

该命令的功能是把由键盘输入的权利交给由第n号插座上的外围设备。但有某些（如自动起动监控的）APPLE Ⅱ计算机无此功能。

例如：*6　CTRL-K

若*0　CTRL-K，就使监控系统回到接受键盘输入的状态。

十二、打印命令

格式：n　CTRL-P　（即同时按下CTRL及P键）

其中：n为插座号n=1～7。

该命令的功能是将屏幕显示的信息输出到连接在第n号插座上的打印机上，并打印出来。如*1插座已接打印机，则*1　CTRL-P。这时打印机打开，并开始打印输出屏幕显示的信息。当*0　CTRL-P则停止输出打印，回到原来的屏幕方式。

自动起动监控的计算机无此功能，而是用PR#1打开打印机，用PR#0关闭打印机。

十三、用户自己定义命令

该命令（CTRL-Y）的功能是强迫监控跳到＄3F8地址（用户程序入口指针）。若用户已将转移命令放在＄3F8指向用户的程序，则当按下CTRL-Y键时，便可执行用户的程序。

例如：若用户的程序入口在＄0C50，欲用CTRL-Y来执行，则*3F8：4C 50 0C，修改指针为JMP ＄0C50。

当按下CTRL-Y时，监控便执行用户的程序了。

十四、十六进制的加减运算（十六进补码运算）

格式：<数值>±<数值>

例如：

```
*20+13
  =33
*3-4
  =FF
*FF+4
  =03
```

第四章　小型汇编及调用监控子程序

前面已谈过，在DOS 3.3系统主盘里，小型汇编的翻译程序将跟随整数BASIC的解释程序，一起被载入16K RAM板（语言板）中，只要输入相应的命令，即可进入汇编语言状态（提示号为“!”），让使用者能较方便地用小型汇编编制机器语言程序。

关于6502指令系统，参看附录四。

第一节　小 型 汇 编

使用小型汇编的必要条件是：APPLE Ⅱ微型机内已有语言板和DOS3.3系统主盘。使用时，先在#1驱动器中插入DOS 3.3系统主磁盘，然后用引导（Booting）命令将上述有关信息引导入16K RAM中。引导命令是：

PR # 6　　或　IN # 6

　　　　　或　* C 600 G

接着，便可通过以下步骤进入汇编状态：

```
]INT
>CALL—151
*F 666 G
!          （可用小型汇编）
```

汇编程序是由程序入口地址引导进入的（见下例）。在程序中，␣表示留一个空格（按一次空格棒）。

```
! 802：LDY␣$801   （按RETURN键）
0302—AC  01  03      LDY   $0301
! ␣LDX   $301  （按RETURN键）
0305—AE  01  03      LDX   $0301
! ␣LDA#$04（按RETURN键）
0308—A9   04       LDA#$04
! ␣JSR   $FCA8   （按RETURN）键
030A—20  A8  FC      JSR   $FCA8
! ␣LDA   $C030    （按RETURN键）
030D—AD  30  C0      LDA   $C030
! ␣INX   （按RETURN键）
0310—E8        INX
! ␣BNE   $310   （按RETURN键）
0311—D0  FD       BNE   $0310
! ␣DEY   （按RETURN键）
0313—88        DEY
! ␣BNE   $305   （按RETURN键）
0314—D0  EF       BNE   $0305
! ␣DEC   $300   （按RETURN键）
0316—CE  00  03      DEC   $0300
! ␣BNE   $0302   （按RETURN键）
0319—D0  E7       BNE   $0302
! ␣RTS      （按RETURN键）
031B  60       RTS
```

这个程序使用$0300及$0301来存放数据，供程序计

数。而＄0302到＄031B这段地址，则用于存放这个程序。只要输入了这个程序，就可以使用下面方法检查它是否正确：

先返回监控状态：！ ＄FF 69 G

后列清单：　　　＊0302 L

屏上即显示出刚输入的汇编源程序（右边）及汇编成的机器语言程序（左边）。

下面试输入以下练习程序，并列清单检查是否出现输入错误，最后执行该程序。

程序入口设在：＄1D00

```
LDX   #$00
JSR   $FB1E
STY   $1CFF
INX
JSR   $FB1E
LDA   $C080
DEC   $1CFF
BNE   $1D0C
LDA   $C080
INY
BNE   $1D14
JMP   $1D00
```

按小型汇编的格式输入以上练习程序后，为慎重起见，还要用列清单（LIST）命令再检查一次：

！ ＄FF69　G

＊1D00　L　　若没有错误，则可执行（调试）。

执行：1D00　G　　若在BASIC状态下，则用CALL 7427来执行。

这是一个发音程序，执行后，如果有游戏操纵杆，便可以控制APPLE Ⅱ发出各种声音了。最后用RETURN键结束该程序。

第二节　小型汇编程序指令的格式

小型汇编程序共使用6502汇编语言程序的56个记忆符(MENMONIC)和13种寻址方式。这些记忆符是标准的，并且与用在MOS Technology/ Synertek 6500 Programming Manual 上的符号一样，只是在寻址方式上稍有不同。

小型汇编程序对地址的解释同监控程序一样，有零页地址（两位十六进数表示）和绝对地址（四位十六进数表示）之分，“$”号表示地址用十六进数字组成。

请读者注意：绝对寻址模式和零页寻址模式的地址，可以无语法上的区别。如果送一个指令给小型汇编程序，它可以用绝对型或者是零页型来解释，那么小型汇编程序就要视其操作数来决定。若其值大于$FF，就用绝对型；若小于$0100，就用零页型。

指令是使用累加器和隐含寻址的，不需要操作数。分支指令使用相对寻址模式。若其操作数的绝对值大于127（在-127和+127之处），小型汇编程序将会发出“哔”声，并随即显示出“∧”，表示有错误产生。

小型汇编程序的寻址方式如下表所示。

模 式 (Mode)	格 式(format)	范 例
累加器(Accumulator)	A	ASLA
立即(Immediate)	# $ {十六进数值}	LDA# $ 3F
绝对(Absolute)	$ {绝对地址}	STA $ 8327
零页(Zero Page)	$ 零页地址	LDA $ 25或 LDA $ 0025
零页变址 (Indexed Zero Page)	$ 零页地址, X $ 零页地址, Y	LDA $ 25, X LDA $ 29, Y
绝对变址 (Indexed Absolute)	$ 绝对地址, X $ 绝对地址. Y	STA $ 0C22, X STA $ 0C33, Y
隐含(Implied)	无	DEX
相对(Relative)	$ 绝对地址	BEQ $ B20
先变址X, 后间接 (Indexed Indirected)	($ 零页地址, X)	LDA($ 3C, X)
间接后变址 Y (Indirec Indexed)	($ 零页地址, Y)	STA($ 5A, Y)
绝对间接 (Absolute Indirect)	($ 绝对地址)	JMP($ 63F0)

第三节　监控子程序的调用

在APPLE Ⅱ机的2K监控程序中，还有80多个可供调用的子程序。它们各自的功能见第四节，可以根据需要调用它们。调用的命令（指令）很简单，用命令：

JSR　入口地址

便可分别调用入口地址所对应的子程序。

参看如下几个范例。

【例1】

```
800  LDA  #$B9     （9的ASCII送A）
     JSR  $FE84    （正常显示）
     JSR  $FDF0    （送字符到屏幕）
     JSR  $FE80    （反相显示）
     JSR  $FDF0    （送字符）
     JSR  $FD8E    （换行、即回车）
     JSR  $FBDD    （"哔"声）
     JMP  $800     （跳回$800入口地址）
```

【例2】　等待一段时间。

```
$900  LDA  #$C1
      JSR  $FDF0
      LDA  #$1F
      JSR  $FCA8    （等待一段时间）
      JMP  $900
```

【例3】　设定并画出低分辨图形的色点及色带。

```
$AOO  LDA  #$0D
      JSR  $F864    （设为（黄色）低分辨图形
```

方法）

```
LDA  #$14   （Y坐标）
LDY  #$0A   （X坐标）
JSR  $F800  （画色块）
LDY  #$14   （水平位置）
LDA  #$02   （顶的位置）
LDX  #$27
STX  $2D    （底的位置）
JSR  $F828
BRK
```

输入后，先退出监控状态，打入GR，进入低分辨图形状态，然后再执行此程序。

第四节 监控程序中一些有用的子程序

若机器语言程序要使用某一监控子程序，那么只要依照该子程序所需的条件，在适当的内存地址或6502微处理器的内部寄存器上，给予必要的值，再用转子指令，使JSR跳到该子程序的入口地址，便可以执行该子程序，实现预定的功能。

下面介绍监控程序中一些有用的子程序。

入口地址	标记符号	功　　能　　注　　释
$FDED	COUT	输出一个字符。先将被输出的字符对应的 ASCII 码预置于累加器内，然后调用 COUT 便可将字符输出；也可以直接用 COUT1子程序。
$FDF0	COUT1	输出一个字符到屏幕上。COUT1 将在累加器内的字符输出到屏幕上的目前输出显示点的位置，并将输出显示点向前移动一格，同时参考正常/反相（Normal/Inverse）地址的设定值，以决定显示的方式。当 COUT1返回时，所有寄存器的值保持不变。
$FE80	SETINV	设定 COUT1用的反相显示模式。所有字符将以白底黑点输出，除寄存器 Y被设定为$3F 之外，其它均不变。
$FE84	SETNOM	为 COUT1设定正常显示模式。所有的输出字符将以黑底白点显示，寄存器除Y被设定为$FF外，其它均不变。
$FD8E	CROUT	产生一个回车字符(返回)。CROUT送出一个RETURN字符到目前的输出设备。
$FD8B	CROUT1	回车及清屏末。CROUT1 由现在的显示点位置起，到文本窗口边为止，然后再调用CROUT。
$FDDA	PRBYT	显示一个16进制的字节（两位十六进制数）。此子程序将累加器的内容（字节），以16进制数的形式送到目前的输出设备，累加器的内容将被改变。

（续上表）

入口地址	标记符号	功　能　注　释
$FDE3	PRHEX	显示一位十六进制数。此子程序将累加器的低四个二进制位（Nybble），以一个位的十六进制数字输出，累加器的内容被改变。
$F941	PRNTAX	将A和X的内容以十六进数显示出来。它以四位十六进制数的形式，将寄存器A和X内的内容输出，累加器A将是第一个字节输出，寄存器X将是第二个字节输出，回复后累加器的内容亦会被改变。
$F948	PRBLNK	印3个空白。将3个空白送到标准输出设备，返回后寄存器A通常含$A0，寄存器X含0。
$F94A	PRBL2	印出很多空白。此子程序输出1～256个空白到标准输出设备上。进入此子程序时，寄存器X内的值表示要输出几个空白，如果X＝$00，那么将输出256个空白。
$FF3A	BELL	输出一个“bell”字符。它输出一个铃响字符（CTRL-G）到目前输出设备，返回后，累加器A含$87。
$FBDD	BELL1	使扬声器发出“Bi”声。此子程序使扬声器以1千赫兹的频率，在0.1秒的时间内发出“Bi”声，寄存器A和X的内容都将改变。
$FDOC	RDKEY	取得一个输入字符。这是标准的字符输入子程序，它将屏幕上闪烁的输入显示点置于目前的输出显示点位置，并且跳至目前输入子程序。其地址存于KSW内（地址$38和$39），通常是KEYIN子程序。

（续上表）

入口地址	标记符号	功　能　注　释
$FD1B	KEYIN	读取键盘。这是键盘输入子程序，它读取键盘，等候按键，并且将一个随机数随机化。当它得到一个按键后，立即消除输入显示点，并将键码置于累加器内。
$FD6A	GETLN	取得一个输入行，并带有即发字符。此子程序集合一个输入行的输入，调用它时必须将适当的即发字符置于地址$33处。它返回时放在输入缓冲区（从地址$200开始）的数据输入行，寄存器X保存输入行内字符的数目。
$FD67	GETLNZ	取得一个输入行。GETLNZ是GETLN的另外一个进入点，它先引起标准输出做回车换行，再进入GETLN。
$FD6F	GETLN1	取得一个输入行，但无即发字符。GETLN1也是GETLN的另一个进入点，它集合一个输入行，但不引发即发字符的出现。如果使用者消除了整个输入行，那么它就将会引发地址$33内的即发字符，并且做另外一行的输入。
$FCA8	WAIT	等待（延时）。此子程序的功能是等待一段特定时间后，再回复到调用它的程序。等待的时间长短由累加器内的值决定。假设A是累加器内的值，等待时间的计算法是：$\frac{1}{2}(26+27A+5A^2)$ 微秒，它返回时A的值为0，但寄存器X、Y不变。

（续上表）

入口地址	标记符号	功　　能　　注　　释
$F864	SETCOL	设定低分辨率图形的颜色。此子程序设定低分辨率图形在屏幕上所用的颜色，其颜色值存于累加器中。
$F85F	NEXTCOL	将颜色值增加3。将目前在用低分辨图形的颜色值增加3。
$F800	PLOT	在低分辨率屏幕上画一方块。此子程序以既定的颜色，在低分辨率屏幕上画一方块。此方块的垂直坐标置于累加器内，水平坐标置于寄存器Y内。它返回后，A值变了，X、Y的值不变。
$F819	HLINE	画一水平行的方块。此子程序以既定的颜色，在低分辨率屏幕上画一水平行的方块。HLINE的垂直坐标置于累加器内，最左的水平坐标置于寄存器Y内，最后的水平坐标置于地址$2C内。它返回时，寄存器A、Y改变，而X不变。
$F828	VLINE	画一水平行的方块。此子程序以既定的颜色，在低分辨率屏幕上画一水平行的方块。HLINE的垂直坐标置于累加器内，最左的水平坐标置于累加器内，最底坐标置于地址$2D内。VLINE返回时，累加器内的值将全部改变。
$F832	CLRSCR	清除整个低分辨率屏幕。如果是在程序文本型调用CLRSCR，则整个屏幕将充满反相型的“@”符号，CLRSCR亦破坏了A和Y的内容。

（续上表）

入口地址	标记符号	功能注释
$F836	CLRTOP	清除低分辨率屏幕的顶。CLRTOP和CLRSCR的功能一样，但CLRTOR只清除屏幕的40行。
$F871	SCRN	读低分辨率屏幕。此子程序同调用PLOT的方式一样，带回低分辨率屏幕上某一方块的颜色值。此颜色值将被存于累加器A内。除A外，其它寄存器的值均不变。
$FB1E	PREAD	读游戏控制器。PREAD会带回代表此游戏控制器位置的数目。在调用此子程序前，必须先把游戏控制器的号码（0～3）置于寄存器X内，如果此号码不实，则将有奇怪的事情发生。PREAD带回的数目在$00～$FF之间，并且置于寄存器Y内。
$FF2D	PRE	印出“ERR”。此子程序将“ERR”及接着的一个铃响（Bell）符号送到标准输出设备，A内的值将被改变。
$FF4A	IOSAVE	保存所有的寄存器。此子程序将6502内部寄存器的值，以A→X→Y→P→S的顺序存入$45到$49的内存地址。A、X的值将被改变，十进位的模式亦被清除。
$FF3F	IORESC	将所有寄存器的值送回。此子程序将地址$45到$49内的值送回到6502的内部寄存器。

第五节　监控程序的特殊地址

第三页监控程序的地址：

<table>
<tr><th colspan="2">地　　址</th><th>用　途</th><th>用　　途</th></tr>
<tr><td>十进制数</td><td>十六进制数</td><td>监控程序ROM</td><td>自动起动ROM</td></tr>
<tr><td>1008
1009</td><td>$ 3F0
$ 3F1</td><td>无</td><td>记处理“BRK”指令的机器语言子程序的地址通常的值为（$FA59）</td></tr>
<tr><td>1010
1011</td><td>$ 3F2
$ 3F3</td><td>无
无</td><td>软进入点向量</td></tr>
<tr><td>1012</td><td>$ 3F4</td><td>无</td><td>开机字节</td></tr>
<tr><td>1013
1014
1015</td><td>$ 3F5
$ 3F6
$ 3F7</td><td colspan="2">记一条转移指令，以跳到FP　BASIC中处理“&”命令的子程序，通常是$4C　$58　$FF</td></tr>
<tr><td>1016
1017
1018</td><td>$ 3F8
$ 3F9
$ 3FA</td><td colspan="2">记一条转移指令，以跳到处理用户命令的子程序。</td></tr>
<tr><td>1019
1020
1021</td><td>$ 3FB
$ 3FC
$ 3FD</td><td colspan="2">记一条转移指令，以跳到处理非屏蔽中断请求的子程序。</td></tr>
<tr><td>1022
1023</td><td>$ 3FE
$ 3FF</td><td colspan="2">记处理中断请求（IRQ）的子程序地址。</td></tr>
</table>

第六节　小　　结

为了叙述的方便，我们约定使用如下符号：

{地址}：存贮器地址

{数值}：数值（数据）

{目标}：目标地址

{首址}：开始地址

{末址}：结束地址

一、监控程序命令的总结

检查存贮器内容：

{地址}↙	检查一个地址内的值。
{地址1}·{地址2}↙	检查在{地址1}和{地址2}之间所有地址单元的值。
RETURN	由最后打开地址开始，显示8个地址的值。

更改存贮器的内容：

{地址}：{数值}{数值}…	将冒号后的值依次由{地址}地址开始存入存贮器内。
：{数值}　{数值}…	由"下个可变地址"开始，依次存入冒号后面的值。

移动和比较：

{目标址}<{首址}·{末址}M↙

将{首址}和{末址}之间的存贮区段，移到由{目标地址}开始的存贮区段。

{目标址}<{首址}·{末址}V↙

	将{首址}和{末址}之间的存贮区段与由{目标址}开始的存贮区段各地址的内容逐个比较。
录音带上数据的存放：	
{首址}·{末址}W↙	将{首址}和{末址}之间所有存贮地址上的值写到录音带上，前面加个10秒长的前导波。
{首址}·{末址}R↙	由录音带上读回数据，存于{首址}和{末址}之间的存贮区。若有错误发生，则印出"ERR"。
执行和列出程序：↙	
{地址}G	执行由{地址}开始的机器语言程序。
{地址}L	分解和显示由{地址}开始的20条指令，接下来的L会显示再下面的20条指令。

二、小型汇编程序（只在INTEGER BASIC上使用）

F666G	进入小型汇编程序
$ 命令	在小型汇编程序内，执行监控程序的命令。
$FF69G	离开小型汇编程序，回到监控程序
{ 地址 } S	单步侦错：分解显示和执行{ 地址 }所指的一条指令，

并同时显示6502内部寄存器的内容，紧接着的S命令，会执行和显示次一条指令（即每一个S命令便执行一条指令）。
注：自动起动监控的APPLE Ⅱ机没有此功能。

{地址}T　　连续侦错：等于连续的S命令，只会在执行到BRK或按RESET键才会停止。
注：只有旧式监控系统才有此功能。

三、其它命令

I　　设定字符反相显示型。

N　　设定正常显示型。

CTRL-B　　冷起动：进入驻留在ROM中的BASIC语言——FP BASIC。

CTRL-C　　中断正在执行的程序或正在执行的LIST命令。当电脑不执行用户程序或LIST命令时，CTRL-C还有另一个用途。见下述。

CTRL-X　　取消现在光标所在的那一行。

CTRL-C　　暖动起：再进入驻留在ROM的BASIC语言——FP BASIC。

{数值}+{数值}	加两个十六进制数，并印出其结果。
{数值}-{数值}	做十六进制的减法，并印出其结果。
{插座号} CTRL-P	将内容输出到{插座号}所指定的设备。若{插座号}=0，则返回正常方式的屏幕输出（自动起动监控的APPLE Ⅱ机无此功能）。
{插座号} CTRL-K	由{插座号}指定的设备输入数据。若{插座号}=0，则返回由键盘输入（自动起动监控的APPLE Ⅱ无此功能）。
CTRL-Y	跳到地址是$3F8的机器语言子程序。

四、各种语言状态之间的转换操作

现将各种语言状态之间的转换操作列表如下(见45页)。

表中仅列出APPLE Ⅱ语言系统（在本机或DOS3.3支持下）之间的转换操作，然而，可在APPLE Ⅱ上使用的语言系统还有很多。例如：

在CP/M系统支持下的还有MBASIC、GBASIC、CP/M-FORTRAN 80等；

在Pascal系统支持下的Pascal语言、FORTRAN 77等。

这些加强的语言系统的进入方法或转换方法可参考有关章节，在此不予详述。

语言状态 从	语言状态 到	操作	条件	注意
]	>	INT	有16KRAM板，并引导DOS3.3	用时要小心！在内存中的程序将全部清除
>	]	FP	有16KRAM板，并引导DOS3.3	
]	*	CALL-151	常规	
>	*	CALL-151	有16KRAM板，并引导DOS3.3	
*	!	F666G	有16KRAM板，并引导DOS3.3	
×	]	CTRL-B(冷起动) CTRL-C(暖起动) 或E000G		冷起动会清掉内存
*	>	CTRL-C		
!	*	CTRL-C或＄FF69G		
!	>	CTRL-C或＄E003		

第五章　FP BASIC 状态

FP　BASIC即浮点BASIC语言，常称为APPLESOFT。它是APPLE Ⅱ微机最基本的高级程序设计语言之一。它的解释程序占10K字节，存放在内存空间$D000～$F7FF中。该解释程序被做成固件（驻留在三片2732或五片2716 EPROM芯片中）插入APPLE Ⅱ主机上。所以一旦开机（或再按一次CTRL-RESET键），必定会进入FP　BASIC语言状态，此时即可用FP　BASIC 语言来编程序了。

FP　BASIC 的状态标志（提示号）是“]”。

一、FP　BASIC 的基本符号及其用法说明

1. FP　BASIC的基本符号

FP　BASIC 的基本符号可分为三大类，编程序时，只允许使用这些符号，而不能使用其它符号。

（1）数字符号：0，1，2，3，4，5，6，7，8，9。

（2）字母符号：A，B，C，D，…，W，X，Y，Z（不包含小写字母）。

（3）标点符号：

.（小数点）	,（逗　号）
;（分　号）	:（冒　号）
?（问　号）	!（感叹号）
'（单引号）	"（双引号）

（4）运算符号：

+（加号）　　　　　　+（正号）

-（减号）　　　　　　-（负号）

*（乘号）　　　　　　)（右括号）

/（除号）　　　　　　(（左括号）

∧（或↑，即乘方号）　　=（赋值号）

+（字符串连接号，形式与加号同）

（5）关系符号：

=（等号，形式与赋值号同）

>（大于号）　　　　>=（大于等于）

<（小于号）　　　　<=（小于等于）

<>（不等于号）

（6）专用符号：

$（字符串符号）

␣（空格号）

2．基本符号的用法说明

（1）乘号必须用“*”来表示，不能用“·”或“×”来代替，更不能将“*”省略。

（2）除号必须用“/”表示，不能用“÷”。

（3）乘方号“∧”（或↑）表示自乘。例如：N^5应写成N∧5。在FP　BASIC解释程序中，N∧5是通过表达式E×P（5*LOG(A)）来计算的。由于对数、指数、函数计算的近似性，故利用乘方计算也具有一定的近似性。例如：3∧4=80.9998，而3*3*3*3=81。因此，当指数是整数时，应采用连乘而不用乘方形式。

（4）在FP　BASIC中只有一种括号“(　)”，因此，

当有多重括号时，可在(　)内再套用(　)。

(5)+(正号)、-(负号)与+(加号)、-(减号)，尽管形式相同，但属不同的运算符。正、负号是对单个数值或数值变量的一元运算，而加、减号是对二个数值变量的二元运算。

(6)只有一个字符串的运算符"+"为字符串的连接符号，它的功能是将字符串常数或字符串变量连接起来。

(7)赋值运算符"="是将一个数值或已给定数值的变量，表达式赋值给一个数值变量。它也可以将一个串常数或已给定串常数的字符串变量赋给一个字符串变量。例如：

```
A=267.5
A$="ABCD"
C$=A$+"EFG"
```

说明：

1)赋值运算符"="，虽然在形式上与等号相同，但其意义与等号不同。赋值运算是不能两边互换的。如A=267.5不能写成267.5=A。

2)使用赋值运算符进行运算时，必须注意数据类型，应将立即数值赋给数值变量，将串常数赋给字符串变量。

(8)字符串常量

用双引号括起来的字符数字串(由基本字符集的字符组成)称为字符串常量。

最长的字符串常量可达255个。

每行可输入的字符可达255个。

二、FP　BASIC中数的表示法

在FP　BASIC中所使用的数，一律采用十进制。它有实数型和指数型两种表示形式，指数型也称为科学记数法。

实数型形式：数中只有数字、小数点和正负号（且正号可以省略）。如：

-846.52　　28.075　　3.14159

指数型形式：它由尾数、E和阶码三部分组成。其中尾数表示数字的有效部分，E表示以10为底数的指数。如：

12.347E+5　　相当于1234700

-9.0702E-6　　相当于-0.0000090702

有效数字（即尾数）：9位

实数的允许范围：10^{-88}～10^{+88}

整数的允许范围：-32767～32767。

三、变量

1．数值变量

（1）简单变量

简单变量可以用字母A～Z中的任一个和任两个符号来表示，也可以用一个字母或两个字母后跟数字0～9中的任何字符来表示，但变量名称不得含有命令中的保留字。

由此可见，APPLE Ⅱ可分辨两个变量字母，但变量名称的长度可达238个字。

（2）数组变量

数组是由某些具有共同特性的数值变量组合而成的。数组内的每个变量称为该数组的数组元素，它们以不同的序数（或称下标）来区分。

FP BASIC数组可以有一维、二维、三维，甚至可多达88维。数组变量用一个字母作为数组名，后跟的下标用括号表示。

如一维数组，一般形式为α（β）。其中α是数组名，它是A～Z中的任一字母；β是下标，它可以是非负的常数、数

值或非负的变量或算术表达式。例如：

N（0），N（1），…，N（12）

N（I）

N（2*J+1）

使用一维数组时，必须注意下列三点：

1）下标β的取值不能为负数；

2）下标β的取值必须为正整数；

3）允许下标的最小值为0。当一维数组所拥有的变量超过11个时，需加语句DIM来说明。

又如二维数组，一般形式为α（β，γ）。其中α为数组名，β和γ分别称为第一、第二下标，且中间用逗号分隔开。

使用二维数组时，必须注意下列两点：

1）二维数组变量的下标取值要求与一维数组相同，下标不取负值，当下标为非整数时，计算机能自动取整。下标的下界为0。当二维数组拥有的变量超过121个时，需用DIM语句加以说明。

2）BASIC解释程序在内存中对二维数组变量是按行存贮的，第一下标是行下标，第二下标是列下标。

例如，有一个三行四列的数组A：

A（0，0）　A（0，1）　A（0，2）　A（0，3）

A（1，0）　A（1，1）　A（1，2）　A（1，3）

A（2，0）　A（2，1）　A（2，2）　A（2，3）

则数组变量在内存中的排列次序为：A（0，0），A（0，1），A（0，2），A（0，3），A（1，0），A（1，1），…，A（2，3）。

2．字符串变量

串变量可以用字母符号A～Z中的任一字母或任两个字母后跟一个串符号＄来表示。

例如：N＄，A2＄，ST＄。

四、运算表达式

运算表达式是通过算术运算符＋、－、＊、／，（　）将数字、已赋值的数值变量或函数连接起来的式子。

例如：－5＊A＋(4＋C/D)是一个算术表达式；"ABC"＋"DEF"＋"13579"是一个字符串表达式。

在FP BASIC算术表达式中，运算规则是：

（1）先乘除，后加减（＊、／优先于＋、－）；

（2）乘方优先于乘、除；

（3）函数优先于乘方；

（4）括号最优先。

第六章　FP BASIC程序命令和基本程序语句

本章介绍FP BASIC的命令和基本程序语句，并安排相应的编程实例，供读者上机练习，以利于掌握FP BASIC程序语句的基本功能，为进行程序设计打下良好的基础。

第一节　立 即 命 令

不需行号引导的语句称为立即（或直接）命令。它的特点是：命令被输入后，按下RETURN键，该命令便被立即执行，并且在执行后立即消失，即不再存贮于计算机内。立即命令有下列六种。

（1）PRINT——打印。例如：

A＝20

PRINT　A

20　（这里显示20，就是立即执行了PRINT A这个立即命令的结果，而PRINT命令已消失了）。

（2）LIST——列表（在屏幕上列出程序清单）。例如：

LIST　（列出全部程序）

LIST　10　（列出第10行程序）

LIST　100～1020　（列出第100行至1020行之间的程序）

（3）RUN——执行程序。例如：

RUN　（从第0行起执行程序）

RUN　100　（从第100行起执行程序）

（4）CONT——继续运行已停止执行的程序

（5）NEW——清除内存中的程序。例如：

NEW

LIST（不会列出程序清单）

（6）DEL——删除指定的程序。例如：

DEL　20，50　（删除第20行至50行的程序）

DEL　100　（删除第100行的程序）

第二节　基本程序语句

有行号引导的语句称为程序语句。其特点是：语句输入后，按下RETURN键，计算机并不立即执行该语句，还需打入RUN命令后才开始执行程序。执行后，程序便保留在计算机中（FP BASIC中行号在0～63999之间）。

程序设计者就是利用程序语句把自己的意图授给计算机，然后通过RUN命令让计算机执行程序的。

下面介绍FP　BASIC的一些基本程序语句：

一、打印输出语句

（1）格式：PRINT <表达式（组）>

　　功能：计算输出表达式（组）

（2）格式：PRINT或简写为"？"

　　功能：打一个空行。

二、赋值语句

格式：LET <变量> = <表达式>

功能：把等号右边表达式的计算值赋给等号左边的变量。

【例1】

```
10  LET  P=3.14159
20  X=P/6
30  Y=SIN(X)+COS(X)
40  PRINT  P, X, Y  (请注意打印项间采用逗号
                      与分号时的区别)
50  END
RUN
3.14159    0.523598    1.36602
```

【例2】字符串变量的一般赋值形式。

```
10  DIM  W$(10), Z$(15)
20  W$="***"
30  Z$="A  BASIC  PROGRAM"
40  PRINT  W$; Z$; W$
50  END
RUN↙
***A  BASIC  PROGRAM***
```

三、键盘输入语句

格式：INPUT <变量组(或INPUT"说明";变量组)>

功能：程序运行到一个INPUT语句时，计算机在屏幕上打印出一个"？"号，然后动态停机，等待用户用键盘按顺序给所有变量逐个赋值，每输入完一个值(数字或字符串常数)，紧跟一个回车(或"，"号)来回答。如果回答的类型不当(如以常数回答字符串变量，或以字符串常数回答数值变量)，则计算机仍等待重新回答。直至回答完，才继续执行下一语句。

四、键盘单字输入语句

格式：GET < 变量或 GET 字符串变量 >

功能：程序运行到一个 GET 语句时，计算机便处于动态停机状态，等待用户用键盘输入一个字符，并赋给GET指定的字符串变量。这是一种很有用的功能，因为它允许使用者在计算机运行中，不必使用 RETURN 键来输入数值。流行的电视游戏，如高炮射击、走迷宫、打网球、田径运动会、拯救组等，就是利用 GET功能来模拟的。

【例 3】

```
10  INPUT  A, B, C$
20  PRINT  "* * *"; A; B; C$
30  INPUT  "VALUES  OF  D, H="; D, H
40  G=D+H/100
50  PRINT  G
60  INPUT  "YOUR  NAME:"; N$
70  PRINT  "WISH  YOU  PROGRESSING!"
80  END
```

试分析此例，并试运行该程序，看应如何赋值？

【例 4】

```
5   PRINT  "IF  OK  KEYIN  'Y'  FOR YES"
10  GET  A$
20  IF  A$  <  >  "Y"  THEN  GOTO  10
30  PRINT  "OK!  LET'S  GO"
40  GET  B
50  IF  B=6  THEN  GOTO  40
60  PRINT  "1984":  PRINT  "Z.C.Z."
```

试分析此程序执行后的情形，并思考按下什么键时程序才结束？

```
10  X= PEEK (-16384)
20  IF X<128 THEN GOTO 10
30  A$=GHR$(X)
40  PRINT A$
50  POKE -16368, 0
```

以上程序与GET A$命令具有同样的功能。

五、成批赋值语句

DATA是读数语句，READ是置数语句，RESTORE是恢复数据区语句（数据指针初始化语句）。

（1）读数语句

格式：DATA 〈数据组〉

功能：为READ语句中的变量组提供一组数据。

说明：

1）数据必须是真正的数值（包括浮点数和定点数）或字符串常数，不能是变量或表达式。数据之间用“，”号隔开，但最后一个数据不能用“，”号。

2）如果数据太多，一行写不完，可以换行另写DATA语句。对于具有多个DATA语句的程序，则数据应视为从左到右、从上到下的一个数据序列，而与DATA语句的多少无关。在源程序输入后，DATA语句中数据已全部依序送入计算机内存的数据区中，因此一组数据无论是用一个DATA或用几个DATA语句给出，也无论DATA语句是处于程序的开头、最后或是其它任何位置，只要数据的顺序不变，其效果是相同的。

3）语句中数据的排列顺序应严格与READ语句中对变量的赋值顺序相一致。

4）DATA语句是一个非执行语句，它并不使计算机产

生任何操作，只是在执行READ语句时，才起用DATA语句所提供的数据，所以在一个程序中有DATA语句时，也必须同时有READ语句。但DATA语句不一定要紧靠READ语句，它可以放在程序中的任何位置。

（2）量数语句

格式：READ〈变量组〉

功能：执行到READ语句时，BASIC解释系统便查找DATA语句，查到后把未用到的数据取出，赋给当前的变量。

说明：

1）变量可以是简单变量、数组变量或字符串变量，变量之间必须用“，”号分隔开。

2）一个程序中可以有多个READ语句，一个READ语句中可以有多个变量，但必须在一行中写完，否则要另写READ语句。

3）READ语句和DATA语句必须联合使用才起作用，为此，它们之间必须符合下面的规则：

①READ语句中变量赋值的顺序必须要与DATA语句中数据排列的顺序一致，并且变量的类型要与数据的类型相符合。例如：

```
10  DATE   1,   "AB", 2, "CD"
20  READ   X, A$, Y, B$
```

②DATA语句中数据的个数不能少于READ语句中变量的个数，否则会因READ语句中变量得不到数据而出错，这是因为READ语句中的各变量从数据区中读取数据是逐个进行的。可以认为在语句的数据表中有一根“指针”，它的初始位置在第一个数据（即程序中标号最小的DATA语句中的第

一个数据）前面。当READ语句中第一个变量需要读数时，就读指针后面的一个数据，读好这个数据后指针就移到第二个数据前面。第二个变量需要读数时，就读指针后面的数据（即第二个数据），依次类推。如果第一个DATA语句（标号最小的DATA语句）中数据多于第一个READ语句中变量的个数，则第二个READ语句中的变量接着读第一个DATA语句中未读过的数据，依此类推。

（3）恢复数据区语句

格式：RESTORE

功能：如上所述，在使用READ/DATA语句时，可以认为系统在数据区中设置了一根读数指针。每读完一个数据后，指针就自动指向下一个数据，也就是说，将用来赋值的数据永远是指针当前所指的数据。如果在READ语句之后要排一个RESTORE语句，则程序执行到RESTORE语句时，指针便重新移至第一个DATA语句中的第一个数据的前面，这种功能称为读数指针初始化。此后，若程序中再次出现READ语句时，将从头开始读数给变量赋值。

在用同一批数据重复给变量赋值的情况下，RESTORE语句将使程序简化，且节省内存。例如，要把1到5的数据分别赋值给变量A至E、M至Q、V至Z，一般程序写法为：

```
10  READ  A, B, C, D, E
20  READ  M, N, O, P, Q
30  READ  V, W, X, Y, Z
100  DATA  1, 2, 3, 4, 5
110  DATA  1, 2, 3, 4, 5
120  DATA  1, 2, 3, 4, 5
```

采用以上程序每次都要重复写出数据，若采用RE-

STORE语句，则程序可以简化为：

```
10  READ  A, B, C, D, E
20  RESTORE
30  READ  M, N, O, P, Q
40  RESTORE
50  READ  V, W, X, Y, Z
100  DATA  1, 2, 3, 4, 5
```

执行第10行语句后，第30行READ语句已无数据可读。但有了第20行语句后，使第30号语句又从1开始读起。第40行语句的作用与第20行语句相同。

重复利用的数据愈多，RESTORE语句的优越性越明显。

此外，RESTORE语句只能把指针移至初始位置，而不能移至任意位置，要想把指针移至任意位置，可配合采用“假读”方法。具体做法是：以程序中不采用的变量重复读数，促使读数指针移至所需的数据前面。例如，上例中，如果A到E以1到5赋值，F到H以3到5赋值，W到Z以2到5赋值，则程序可写成：

```
10  READ  A, B, C, D, E
20  RESTORE
30  READ  T, T
40  READ  F, G, H
50  RESTORE
60  READ  T, W, X, Y, Z
70  PRINT  A, B, C, D, E, F, G, H, W, X,
    Y, Z
80  DATA  1, 2, 3, 4, 5
90  END
```

```
RUN
1          2          3          4
5          3          4          5
2          3          4          5
```

由此可见，T是用来作“假读”的变量，它的作用仅仅是为了使后面的变量跳过一个或几个数据去读取所需的数据。

六、无条件转移语句

格式：GOTO〈行号〉

功能：无条件地转去执行GOTO语句指示的那行语句。

【例5】计算并联电路中的电流。

```
5  PRINT "R1, R2, R3, R4, U=";
10 INPUT R1, R2, R3, R4, U
20 LET I=(1/R1+1/R2+1/R3+1/R4)*U
30 PRINT
40 PRINT "I="; I
50 GOTO 5
60 END
RUN
R1, R2, R3, R4, U=? 100, 150, 240, 360, 24
I=0.5667
R1, R2, R3, R4, U=?
```

说明：

1）程序执行到第50行语句时，再不按标号顺序往下执行，而返回执行第5行语句，并按顺序执行下去，即再执行第10行INPUT语句……。这样处理可以实现重复计算，而避免使用运行命令（RUN）。

2）这是一个一旦运行起来就无限地循环下去的程序，

要使它停止运行，就必须使用 RESET 键。

【例6】打印一张课程表。

```
10  DIM  A$(12), C$(2)
20  PRINT  "COURSE", "HOURS", "GRADE"
30  READ  A$, H, C$
40  PRINT  A$, H, C$
50  GOTO  30
60  DATA  "ENGLISH", 3, "A1", "PHYSICS"
70  DATA  5, "B2", "STATICS", 3, "A1"
80  DATA  "COMPUTER", 5, "A1", "HISTO-
    RY"
90  DATA  3, "B2", "HUMANITIES", 3,
    "C3"
100 DATA  "CALCULUS", 2, "C3"
999 END
RUN
COURSE        HOURS       GRADE
ENGLISH         3           A1
PHYSICS         5           B2
STATICS         3           A1
COMPUTER        5           A1
HISTORY         3           B2
HUMANITIES      3           C3
CALCULUS        2           C3
```

七、循环语句

在科学技术和生活领域里，常常遇到一些需要大量重复计算的问题，对这类问题，人们所关心的是如何减少一些不

必要的重复计算，以节省存贮单元和提高计算效率。如打印一张三角函数表，假定要求每隔1°印一个函数值，若用LET语句来写程序，语句的数量就会很多，使程序显得庞大繁琐。如果改用上面讲的GOTO语句来写，则程序如下：

```
10  LET  X=0
20  PRINT  SIN(X), COS(X), TAN(X)
30  LET  X=X+3.14159/180
40  GOTO  20
50  END
```

仅用5条语句，就达到计算三角函数的目的，程序大大简化了。这个程序虽然是一个可执行的程序，但用这样的循环来进行计算，显然有一个致命的缺点，即若不进行人工干预（如按RESET键来中断），程序绝不可能执行到50行语句便停止运行，而是不停地运行下去，形成一个死循环状态。为解决这一问题，BASIC语言提供了循环语句，用以有效地解决重复性的计算问题。上例若用循环语句，则可以写成下列程序：

```
10  FOR  X=0  TO  3.14159  STEP
    3.14159/180
20  PRINT  X, SIN(X), COS(X), TAN(X)
30  NEXT  X
40  END
```

1. 循环语句的格式及功能

格式：FOR‹循环变量›=‹e_1›TO‹e_2›STEP‹e_3›

‹循环体›

NEXT‹循环变量›

功能含义：

（1）循环语句是为实现重复计算而设置的，FOR 语句和 NEXT 语句必须成对出现。若缺 NEXT，则只能循环一次；若缺FOR，则会出错。

（2）e_1 是循环变量初值，e_2 为循环变量终值，e_3是循环变量每次的增量（也称步长）。e_1、e_2、e_3可以是具体的数值，也可以是已赋值的变量或表达式。e_1、e_2的值可以为正、负或零，e_3不能为零，否则循环无止境。

$$\text{循环次数}=\left[\left(\frac{e_2-e_1}{e_3}\right)+1\right] \quad \text{（取整）}$$

（3）当程序执行到 FOR 语句时，BASIC 系统先将初值（e_1）赋给循环变量（例如X），并记下终值（e_2）和步长（e_3），然后把循环变量与终值比较。

当步长为正，且初值小于终值时，则执行循环体中的语句。在此过程中，循环变量一直保持不变，直到程序执行到NEXT语句时，BASIC系统自动地把循环变量的值增加一个步长，然后再与终值比较。如果仍小于终值，则返回执行循环体中的语言，重复以上过程，直到循环变量大于终值，跳出循环而执行紧接NEXT后边的语句。

当步长为负，且初值大于终值时，循环变量由大向小变化，直到循环变量小于终值时，才跳出循环而执行NEXT后边的语句。

当步长为零时，循环变量始终不变，因而循环无止境而形成死循环。

当步长为正而初值大于终值，或者步长为负而初值小于终值时，循环变量变化的方向都会越来越远离终值，也会形成死循环。

（4）程序可以按照一定的条件跳出循环体而直接执行

NEXT语句或循环体外某一语句。但一般来说，不可由循环体外随意跳至循环体中，因为这时循环变量无确定的值。

2。循环语句流程图

根据上面所讲的循环语句的功能含义，我们把循环语句流程图画成如图6－1所示的形式。采用这种流程图来表示循环语句，优点在于比较直观，容易理解，层次清楚。

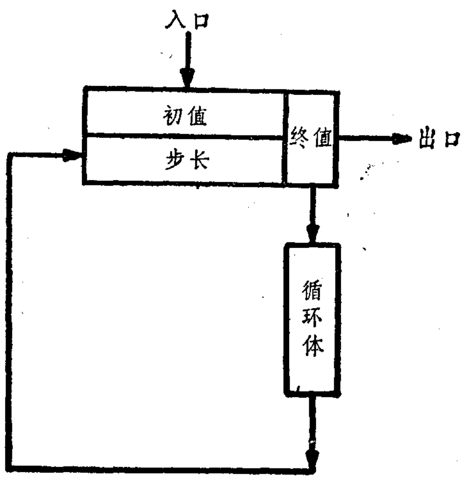


图 6-1 循环语句流程图

3.程序示例

【例7】 初值、终值或步长变化时，各类程序的示例。

（1）步长为正数。

```
10  FOR  X=1  TO  10  STEP  2
20  PRINT  X
30  NEXT  X
40  END
```

```
RUN
1    3    5    7    9
```

说明：因为步长为 2 ，当执行五次后，循环变量X 变为11，大于终值，跳出循环，而执行第40行结束语句。

（ 2 ）步长为 0 。

```
10  FOR  X=1  TO  2  STEP  0
20  PRINT  X;
30  NEXT  X
40  END
```

如果现在按下RUN键，屏幕上会不停地印出“ 1 ”字。若把第20句后的“；”换成“，”或不取任何符号，请读者考虑，这时情况会变成怎样？

说明：因为步长为 0 ，所以循环变量永远小于终值，跳不出循环而形成死循环。

（ 3 ）步长为负数。

```
10  FOR  X=5  TO  1  STEP  -1
20  PRINT  X;
30  NEXT  X
40  END
RUN
5    4    3    2    1
```

说明：由于步长为负，初值大于终值，因此只有当满足循环变量小于终值时才跳出循环而执行下一行（第40行）结束语句。

（ 4 ）初值、终值、步长都是负数。

```
10  FOR  X=-100  TO  -200  STEP  -20
20  PRINT  X;
```

```
30  NEXT  X
40  END
RUN
-100  -120  -140  -160  -180  -200
```

说明：六次循环后，循环变量X成为-220，小于终值，因此中止循环。

（5）步长为1。

```
10  LET  X=1
20  FOR  I=1  TO  5
30  PRINT  X;
40  NEXT  I
50  END
RUN
1    1    1    1    1
```

说明：步长为1，STEP可以省略不写。

（6）循环变量与循环体内变量不一致。

这时循环变量仅起计数作用，因此下面两个程序的作用是一致的。读者试比较之。

程序（a）

```
10  FOR  I=1  TO  5
20  READ  X
30  PRINT  X;
40  NEXT  I
50  DATA  1,2,3,4,5
```

程序（b）

```
10  FOR  I=6  TO  10
20  READ  X
```

```
30  PRINT  X;
40  NEXT  I
50  DATA  1,2,3,4,5,
```

八、注释语句

格式：REM〈注释内容〉

注释语句是一种非执行语句，它对程序的执行不产生任何影响，只是在写程序时为程序或程序中某一部分作注释用。例如：

```
150  REM  FIND  THE  MAXIMUM  VALUE
```

（找最大值）

注释语句REM（REMARK）同样要写行号，它后面的内容可写任何字符，不需加引号，只要全行不超过255个字符就行了。程序执行时屏幕上没有任何反应，只是在命令计算机列出程序表（LIST）时才出现。

注释语句可放在程序中任何位置。它虽不被执行，但占用一些内存单元，所以也要简明扼要。

九、暂停语句

格式：STOP

计算机执行到这个语句时，就会暂停程序的执行，等待下一个命令，并发出“哔”的声音，在屏幕上显示出：

BREAK IN ××

其中××是行号。在暂停期间，使用者可以检查或记录下一个程序段的执行情况，或者打入直接命令。

若程序中有：

```
120  STOP
```

当执行到这句时，屏上显示

BREAK IN 120

要继续把程序执行下去，可从键盘打入命令：

“CONT”并按“RETURN”键。

十、结束语句

格式：END

这个语句可使计算机结束程序的执行并返回到命令状态。

必须指出，BASIC语言一般要求程序最后的一个语句必须为END，但APPLE Ⅱ微型机无此要求，当一个程序执行完毕时，机器会自动回到命令状态。

十一、变量清除语句

格式：CLEAR

如果需要把已经赋值的变量全部清除为零。那么，可以在程序中编入变量清除语句。

【例8】

```
10  A=1:B=2:C=3
20  PRINT  A,B,C
30  CLEAR
40  PRINT  A,B,C
50  END
RUN
 1          2          3
 0          0          0
```

可见在CLEAR语句后，各变量都清除为零了。

CLEAR不单把所有变量都清除为零，而且还可以把循环和转子程序语句FOR和GOSUB状态重新置定。

【例9】试比较STOP语句及END语句的作用。

```
10  INPUT  A%
```

```
20  IF  A%  >  0  THEN  GOTO  40
30  STOP
40  PRINT  SQR(A%)
50  END
```

提示：试输入0或输入一个负数，观察及监听计算机的运行状态；若把第30行的STOP换成END，再输入一个正数，继续以上观察。观察时，要特别注意：程序运行的功能是否正常，屏幕印出什么信息和响声变化等。

【例10】求1＋2＋…＋n之和（求累加和）的程序。

```
60  S=0
70  INPUT  "N=";  N
80  FOR  I=1  TO  N
90  S=S+I
100  NEXT  I
110  PRINT  "SUM=";  S
```

本例省略了STOP或END语句，试看计算机是否会自动停止。

十二、清屏并归位语句

格式：HOME

HOME语句用于清除屏幕上目前的信息，并把光标移到开始的位置（屏幕的左上角）。HOME语句不清内存。

十三、置正文显示方式语句

格式：TEXT

TEXT是置屏幕显示为文本方式的语句，在文本方式下，使用者可以将输入的程序显示在屏幕上。通常，打开APPLE Ⅱ时，计算机会自然设定为文本方式的。正如上述所做的程序练习，就是在文本方式中进行的。

十四、置反转显示方式语句

格式：INVERSE

INVERSE语句是使屏幕显示的大写字母或图形由正常状态（黑底白字），转变成反转（反相）显示（白底黑字）状态。它使显示方式多样化，如果要特别指出某点显示内容，便可选用反转显示方式。

十五、置闪烁显示方式语句

格式：FLASH

十六、置正常显示方式语句

格式：NORMAL

NORMAL可将目前的反转显示方式恢复为正常显示方式。通常，APPLE Ⅱ开机时，会自然置为正常显示方式的。

【例11】显示方式的选择。

```
5  HOME
10 TEXT
15 A$="THIS  IS  AN"
20 B$="APPLE  II"
30 C$="MICROCOMPUTER"
40 PRINT  A$
50 INVERSE
60 PRINT  B$
70 NORMAL
80 PRINT  C$
90 PRINT  "10,January,1984"
```

请读者试运行上例程序，并记下运行结果。

十七、程序跟踪与不跟踪语句

格式：TRACE和NOTRACE

TRACE语句是置计算机对所运行的程序进行跟踪侦错，并把所运行的程序行号及运行结果均显示在屏幕上。NOTRACE是结束（取消）TRACE功能的语句。见下例。

【例12】跟踪与不跟踪程序如下：

```
5  TRACE
10 FOR  E=5  TO  10
20 PRINT  E,E*2
30 NEXT  E
40 NOTRACE
50 FOR  E=5  TO  10
60 PRINT  E,E*2
70 NEXT  E
80 END
RUN
#5   #10   #20    5    10
     #30   #20    6    12
     #30   #20    7    14
     #30   #20    8    16
     #30   #20    9    18
     #30   #20   10    20
     #30   #40    5    10
                  6    12
                  7    14
                  8    16
                  9    18
                 10    20
```

十八、输出显示速度的控制语句

格式：SPEED＝〈X〉

APPLE Ⅱ显示信息的输出速度可由SPEED＝X语句进行控制，X在0～255之间，0为最慢，255为最快。

【例13】

```
10  SPEED= 5
20  PRINT "SPEED  TEST  5"
30  SPEED=254
40  PRINT "SPEED  TEST  254"
50  GOTO  10
```

读者试执行以上程序，观察两种不同速率显示的差异。

综 合 例 题

1. 对两个随机数取整求和。

```
10  DIM  A$(25), B(25)
20  PRINT "* * * * * * * * * * * * * * * * *
      * * * * * * * * * * * * *"
30  PRINT" *
                *"
40  PRINT  "*SIMULATING THE  THROWING
            *"
45   PRINT  "*
                  *"
50  PRINT  "*  OF  A  PAIR  OF  DICE
               *"
60  PRINT  "*
               *"
```

```
70  PRINT "********************
    **************"
80  PRINT
90  A$="IF  X,Y,IS:"
100  B$="THEN  SUM  IS:"
110  FOR  I=1  TO  10
120  X=INT(RND(1)*6)+1
130  Y=INT(RND(1)*6)+1
140  A=X+Y
150  PRINT  A$;X;",";Y;"  ";B$;A
160  NEXT  I
170  END
```

2．确定透镜的焦距。

设U为物体到透镜的距离，V为成像屏到透镜的距离，F为焦距。根据成像公式：

$$F=\frac{UV}{U+V}$$

对每一组U和V都可算出F，并可算出F的平均值。程序如下：

```
10  REM  TO  DETERMINE  THE  FOCAL
    LENGTH  OF  A  CONVEX  LENS
20  LET  S=0
30  FOR  I=1  TO  10
40  READ  U, V
50  F=U*V/(U+V)
60  S=S+F
70  NEXT  I
```

```
80  F=S/10
90  PRINT "AVERAGE  FOCAL  LENGTH="
    ; F; "CM"
100  STOP
110  DATA  8.5, 36, 20.44, 31.2, 22.5, 27.5,
     144.44, 30
120  DATA  15.5, 63.2, 24.4, 24.5, 29.5, 20.8,
     36.6, 13.5
130  DATA  19.5, 33.1, 16.3, 46
140  END
```

3．用梯形法求定积分$\int_1^2 X\ \sin(X)dx$的值。

取N等于10，即划分10个小梯形。

程序如下：

```
10  LET  A=1
20  LET  B=2
30  Y=0
40  N=10
50  H=(B-A)/N
60  FOR  I=0  TO  N-1
70  G=A+I*H
80  C=G*SIN(G)
90  P=A+(I+1)*H
100  D=P*SIN(G)
110  S=H*(C+D)/2
120  Y=Y+S
130  NEXT  I
```

```
140  PRINT  "Y=";  Y
150  END
```

4．计算标准偏差。

已知计算公式为：

$$D=\sqrt{\frac{\sum(X-\overline{X})^2}{N}}$$

式中　N为数据个数；

$\overline{X}$为数据平均值；

D为标准偏差。

程序如下：

```
10  REM  CACULATION  OF  STANDARD
    DEVIATION
20  DATA  4, 5, 17, 12, 15, 11, 12, 7, 14, 21
30  LET  T=0
40  FOR  I=1  TO  10
50  READ  X
60  T=T+X
70  NEXT  I
80  A=T/10
90  V=0
100  RESTORE
110  FOR  I=1  TO  10
120  READ  V
130  V=V+(V-A)*(V-A)
140  NEXT  I
150  D=SQR(V/10)
```

```
160 PRINT "D=", D
170 END
```

习题与思考题

1．说出下列命令的功能：

NEW CLEAR DEL LIST RUN TEXT

2．TRACE有何作用？什么语句能解除TRACE状态？

3．指出三个可以设定/输入变量值的语句，并指出它们各自的长处和短处。

4．试述READ/DATA语句的特点。在什么情况下采用READ/DATA语句？

5．下列语句中哪些是错误的？

（1）
```
10 READ A, B, C, D
20 READ E$, F, G%
30 DATA 1, 2, 3, 4/5, SIN(3.14/6),
   OK, 256
```

（2）
```
10 DATA "BOY", "GIRL", 9999
20 READ A$, B, C$
```

6．写出下列程序运行的结果：

```
5  DIM C$(5), G$(5)
10 READ A, B, C$
20 DATA 1, 2, "AB"
30 READ C, D
40 RESTORE
50 DATA 4, 5, 6, 7
60 READ E, F, G$, H, I, J, K, L, M,N
```

```
70  RESTORE
80  DATA  8, 9, 10
90  PRINT  A, B, C $, C, D, E, F, G$,
    H, I, J, K
100 PRINT  L, M, N
110 END
```

7. 以1980年我国工业产值为100，按每年平均增长率为5%，8%，10%，14%计算，试编写程序，分别计算到2000年时的工业产值。

计算公式为：$X = 100 \times (1 + Y)^N$

式中 Y 为年增长率，N 为年数。

8. 分别用READ/DATA语句和INPUT语句，编写四个学生的平均成绩的程序。

计算公式为：$G = \dfrac{5A + 4B + 3C + 2D}{A + B + C + D}$

其中，A、B、C、D分别是取得5分、4分、3分、2分的次数，数据如下。

成绩 / 学生	A	B	C	D
甲	5	6	4	2
乙	6	3	2	1
丙	4	3	2	1
丁	3	4	5	1

第七章　FP BASIC 的其它语句

第六章较详尽地介绍了FP　BASIC最基本的一些语句及其简例，目的是从引导读者入门的角度来介绍FP　BASIC程序设计。本章将进一步介绍其它语句，但叙述将简短些，而例子将全面些，以提高读者的学习效率。

一、水平移动光标语句

水平移动光标语句亦称为行位置自由打印语句。

格式：TAB(〈X〉)

功能：从目前光标所在的行的最左边算起，把光标移到第X格的位置上（X在0～255之间取值，0～39格在本行上，而40～79格在F一行上，以此类推），TAB(〈X〉)仅用在PRINT语句中。

例如：PRINT　TAB(23)　"*"

功能是：把光标移到第23格的位置上，再印出一个"*"符号。

二、垂直移动光标语句

格式：VTAB(〈Y〉)

功能：将光标移到第Y行（Y在1～24之间取值，屏顶为第1行，屏底为第24行）。

例如：VTAB(15)

功能是：将光标移到第15行。

三、本行移动光标语句

格式：HTAB(〈X〉)

功能：将光标移到第X列（X在1～40之间取值）。

四、相对定位打印语句

格式：SPC（‹X›）

功能：从目前光标所在的位置算起，把光标向右移，跨过X个空格再定位。SPC(‹X›)由PRINT引导,SPC(‹X›)后面跟有分号和打印项。X在0～255之间取值。

例如：PRINT SPC(18)；"* * *"

功能是：将光标向右移，跨过18格，再印出 * * *。还可以将几个SPC(X)连在一起使用。

例如：PRINT SPC(155)SPC(250)SPC(100)

也可以在印出一项之后，紧接着使用SPC(X)语句，但两者之间须加分号。

例如：PRINT TAB(10)；"* * *"；PRINT SPC（3）；"OK！"

这例句的功能是：从当前光标所在行的最左边算起，把光标移到第10格位置上，印出 * * *，接着留出3个空格，再印出OK！。

五、取得目前光标的水平位置语句

格式：POS(‹X›)

X可为任意值（无直接意义，只要FP BASIC能够接受即可），通常取POS（0）。POS读得的光标位置是从0起计算的（0～39）。

至此，请读者注意VTAB(‹X›)，HTAB(‹X›)，TAB(‹X›)的位置都是从1算起的；而SPC(‹X›)，POS(‹X›)则是从0开始计算的。

试键入：PRINT TAB(23)；POS（0）

会印出“22”

而键入：PRINT　SPC(23)；POS(0)

则会印出“23”

再来比较：

```
5  HOME
10 A=4:B=7.8
20 PRINT  A; TAB(A); A; TAB(B); B
30 PRINT  A; SPC(A); A; SPC(B); B
```

键入RUN后，请看屏幕显示，有什么情况出现？

【例1】利用行位置自由打印语句，可以打印很多曲线或图案，试运行下面的程序：

```
5  HOME
10 FOR  X=1  TO  8
20 PRINT  TAB(9-X)
30 FOR  A=1  TO  2*X-1
40 PRINT  "*";
50 NEXT  A
60 PRINT
70 NEXT  X
80 END
```

读者可以先设想一下将会看到什么图案，然后输入RUN命令，观察屏幕上显示的图案。

【例2】画一条普通曲线（注意：这里还是在文本方式下画的）。

输入以下程序：

```
10 HOME
20 FOR  X=1  TO  6
30 HTAB  X:VTAB  X
```

```
40  PRINT  "*"
50  NEXT  X
60  END
```

键入RUN命令后，屏幕上将显示出一条曲线。

六、调用机器语言子程序的语句：CALL‹X›

格式：CALL‹X›

其中，X为机器语言子程序入口地址（十进制），X可以是正值（按正向——由小到大计地址数）；X也可以是负值（按负向——由大到小计地址数）。

功能：（详见附录一）

七、条件等待语句：WAIT‹X›，‹Y›，‹Z›

格式：WAIT‹X，Y，Z›

其中，X为十进制接口地址或设备位置。Y为数值（Y在0～255之间取值），Z为数值（Z在0～255之间取值）。

功能：（详见附录一）

八、内存取语句：POKE‹X›，‹数值›；PEEK‹X›

格式(1)：POKE‹X›,‹数值›

功能：把指定的数值存入指定的地址X中（X为十进数地址，数值为十进数）。

例如：POKE 4096，55

功能是：把55存入地址4096中

格式(2)：PEEK‹X›

功能：把地址X的内容取出（X为十进数地址，它的内容也是十进数），PEEK还可以直接赋值。

例如：B＝PEEK（4096），即取出地址4096的内容，并赋值给变量B。

九、传数及执行机器语言子程序语句：USR(N)

格式及功能详见附录一。

十、存贮空间定限语句：HIMEM：及LOMEM：

详见附录一。

十一、查询可用的内存量语句：FRE(0)

详见附录一。

【例3】编程范例

```
10  HOME
20  HIMEM:38400
30  LOMEM:12291
40  TEXT
50  CALL  -  1052
55  M=FRE(0)+(FRE(0)<0)*65535+1
60  PRINT  "FREE  MEMORY=";  M
110  B=FRE(0)+(FRE(0)<0)*65535+1
120  PRINT  "FRE  MEM  IS:";  B
130  CALL  -  1052
140  FOR  I=0  TO  12  STEP  0.01
150  CALL  -  1052
160  NEXT  I
170  GOTO  10
```

试分析、执行上例程序，并察听结果。读者可以设想对上述程序作些修改，使程序运行的效果更有特色。

以下介绍数组和字符串函数。

十二、数组定义（说明）语句

我们已经对数组及其变量的基本概念和规定有所了解。在上一节循环语句中也介绍过数组的简单应用。然而，对于超过11个元素的一维数组和超过121个元素的二维数组，在

使用之前必须用语句加以说明。为了在程序设计中更有效地使用数组，下面将对数组说明的有关规定和用法，进一步作系统的介绍。

1．数组概念的进一步解释

在 BASIC 语言中，数组是一批带下标的变量的集合，而不能象通常理解的那样，把它看成是一批数的集合。这就是说，数组中的每一个元素均为一个变量。对具有确定下标的一个数组元素来说，它和简单变量的作用是基本相似的。数组变量的特点，在于允许有可变的下标。这样，变量也就可以随下标的变化而变化，从而有可能依照下标的变化对一批变量顺序地加以处理，这就为程序设计带来了极大的方便。如果不借助于数组变量，要对上述例 3 类型的问题进行程序设计，那么其繁琐程度往往是不堪设想的。

2．数组说明语句

在一个程序中使用数组变量不象使用简单变量那样省事。对于简单变量来说，只要按规定在程序中正确书写变量名，BASIC 解释系统就能随时给出存放相应变量值的存贮单元。但一般说来，在使用数组变量之前，必须通过数组说明语句向 BASIC 解释系统提供所用数组的有关信息（数组名字、维数、大小），以便系统为数组分配存贮其各变量值的存贮空间，同时系统记录下数组的有关信息，以便随后实现对数组变量的正确访问。下面介绍实现这些功能的数组说明语句。

数组说明语句的格式和功能如下：

格式：DIM　若干数组

其中：数组可以是一维的，也可以是二维的，甚至可以是多达88维的。一个 DIM 语句中数组的个数以写满一行为限，

各数组之间必须以“，”号分开，语句的最后不能有任何标点符号。

功能：

（1）本语句是说明语句，不是执行语句。它仅向BASIC解释系统提供有关数组名字、维数和下标界等信息，以便系统地为各数组元素留下存贮单元。此后，程序就可以通过下标访问这些单元，实现对数组变量值的存取。

（2）系统允许说明可变界的数组，也就是说，在说明语句中可以用取非负整数值的变量作为下标界，但在程序中必须于说明语句前给相应的可变下标界赋值。

程序示例

【例4】设某单位有120名职工，试编写一个程序把他们的年龄，参加工作的日期和工资数分别存入三个一维数组中，并把工资总数打印出来。

以下简单程序便可实现这种要求。

```
10  DIM   A(120), D(120), G(120)
20  LET   S=0
30  FOR   I=1  TO  120
40  INPUT  A(I), D(I), G(I)
50  LET   S=S+G(I)
60  NEXT  I
70  PRINT  "S=";  S
99  END
```

从这个简单的例子可以看到，利用数组来实现对大量数据的存贮和处理是十分有效的。其优越性在今后的程序设计中会逐渐显示出来。在这里，应强调指出，数组的表示形式，如A(120)，当它出现于DIM语句时，作为数组说明

符，表示数组共包含121个元素（下标值是从 0 开始算的）。同样的形式如果出现在其它语句（执行语句）中时，它只代表一个确定的数组元素，A（120）只代表数组中第120个元素。

【例 5 】把两组数

$$x=1, 2, 3, \cdots, m$$

$$y=1, 2, 3, \cdots, n$$

的交叉乘积存入一个二维数组中并打印出来，然后保持数组的大小不变，对数组重新说明后再打印出来。试分析比较两次所得结果。

这种交叉乘积很容易由双重循环来实现，我们给出如下程序，读者可设 $m=3$，$n=4$，察看运行结果。

```
10  INPUT "M, N="; M, N
20  PRINT
40  DIM A(M, N)
50  FOR X=1 TO M
60  FOR Y=1 TO N
70  A(X, Y)=X*Y
80  PRINT A(X, Y),
90  NEXT Y
100 PRINT
110 NEXT X
120 DIM B(N, M)
130 FOR I=1 TO M
140 FOR J=1 TO N
150 PRINT B(I, J),
160 NEXT J
```

```
170  PRINT
180  NEXT  I
199  END
```

本例首先提供了变界数组的用法，由第10句语句来实现对下标的赋值。

十三、数字（符号）变字符串语句

格式：STR$（X）

功能：将括号中的数字或符号转换成相应的字符串。

【例6】

```
10  A=85.7:B=-85.7
20  PRINT  STR$(A)
30  PRINT  STR$(B)
40  PRINT  STR$(A+B)
50  PRINT  STR$(A)+STR$(B)
RUN
85.7
-85.7
 0
85.7-85.7
```

十四、字符串变成数字（符号）语句

格式：VAL(A$)

功能：VAL(A$)是STR$(X)的反变换：把括号中的字符串转换成相应的数字或符号。

例如：若A$="12"，B$="34"

则VAL（A$+"·"+B$），就得数值12.34。

又如：VAL（A$+"E"+B$）得到数值 12E34即12×10^{34}。

但有例外情况：当括号中的字符串含有数字及字符符号，且数字在前而字符在后时，VAL只用字串前面的数字，而不管数字后面的字符符号。

例如：VAL("10000DOLLARS")，就得到10000。

用这种方法可以方便地检索出地址（见例7）。

【例7】

```
10  REM  "WHAT  SIDE  OF  THE  STREET? "
20  REM  EVEN=NORTH,  ODD=SOUTH
30  INPUT  "ADDRESS: NUMBER  AND
    STREET";  AD$
40  C=INT(VAL(AD$)/2)*2
50  IF  C=VAL  (AD$)  THEN  PRINT
    "NORTH  SIDE":GOTO  30
60  PRINT  "SOUTH  SIDE":GOTO  30
```

十五、ASCII码变成字符语句

格式：CHR$(X)

其中，X是该字符的ASCII码，X在0～255之间取值。

功能：将ASCII转换成相应的字符。

十六、字符变成ASCII码语句

格式：ASC(A$)

功能：求出括号内字符串的第一个字符的ASCII码。

例如：以下三个语句将印出同样的数字（ASCII）。

```
10  PRINT  ASC  ("G")
20  PRINT  ASC  ("GRA")
30  T$="GND":PRINT  ASC  (T$)
```

【例8】字符与ASCII码变换的程序：

```
10  HOME
```

```
20  FOR  I=30  TO  255
30  PRINT  I, CHR$(I)
40  NEXT  I
50  STOP
```

十七、求字符串长度语句

格式：LEN(A$)

功能：求指定字符串的字符长度。串变量，串表达式或串常数必须放在括号内。

【例9】

```
10  A$=" "
20  B$="DATA"
30  PRINT  A$, B$, B$+B$
40  PRINT
50  PRINT  LEN(A$), LEN(B$), LEN
    (B$+B$)
DATA        DATA        DATA
  0           4           8
```

十八、左取字符串语句

格式：LEFT$(A$, X)

功能：取出字串A$的最左边的X个字符，X可以是算术表达式。

十九、右取字符串语句

格式：RIGHT$(A$, X)

功能：取出字串A$的最右边的X个字符。

二十、中间取字符串语句

格式：MID$(A$, X, Y)

功能：取出字串A$第X个字符算起的Y个字符（作为

子串）。

【例10】 说明编码/译码的程序

```
5   HOME : TEXT
10  CLEAR
20  INPUT "ENTER MASSAGE"; M$
30  FOR K = 1 TO LEN(M$)
40  T$ =MID$(M$, K, 1)
50  CD=ASC(T$)+5
60  IF CD>255 THEN CD=CD-255
70  NU$ =NU$ +CHR$(CD)
80  NEXT K
90  PRINT "THE CODED MASSAGE IS"
100 PRINT NU$
110 FOR K=1 TO LEN(NU$)
120 T$ =MID$(NU$, K, 1)
130 CD=ASC(T$)-5
140 IF CD<0 THEN CD=CD+255
150 OLD$ =OLD$ +CHR$(CD)
160 NEXT K
170 PRINT "THE DECODED MESSAGE IS"
180 PRINT OLD$
```

程序中30～80句和110～160句，说明了怎样“截取”一个字符串中的字符。语句50、60和150，说明了ASCII代码的操作。读者可试运行此程序。

【例11】字符串检索范例

使用内部的字符串函数MID$和LEN，很容易建立一

个非常灵巧的字符串处理子程序 INSTRING。它的功能是取两个字符串自变量，判断其中一个是否包含着另一个。当要在一大批数据或文字中检索一个特殊的词、短语或一段数据时，它就非常有用。

下面是这一子程序：

```
1000  FOR  I=1  TO  LEN(X$)-LEN
      (Y$)+1
1010  IF  Y$=MID$(X$, I, LEN(Y$))
      THEN  RETURN
1020  NEXT  :I=0:RETURN
```

要使用这个子程序，首先要赋给X$一个长字符串值(即"检索范围")，赋给Y$一个需要的值，然后用 GOSUB 调用子程序。子程序将得到一个 I 值，它告诉使用者 Y$ 在长字符串X$中的起始位置，如果Y$不是X$的一个子串，那么 I 取值为 0。

下面是一个应用 INSTRING 子程序的程序举例（将上面的子程序1000～1020语句加在80语句后）

```
5   CLEAR
10  INPUT "ENTER THE LONGER STRIN G";
    X$
20  INPUT "NOW  ENTER  THE  SHORTER
    STRING", Y$
80  GOSUB  1000
40  IF  I=0  THEN  70
50  PRINT  Y$; "IS  A  SUBSTRING  OF";
    X$
55  PRINT  "ENDING  POSITION:", I
```

```
60  PRINT  "STRATING  POSITION:",I-LEN
    (Y$)-1
65  PRINT:PRINT:GOTO  10
70  PRINT  Y$$; "IS  NOT  CONTAINED
    IN"; X$
80  GOTO  10
```

读者试运行此程序，输入被检索的字符串，然后输入需要的子串。

二十一、条件转移语句

1．语句格式和功能

格式：

(1) IF 〈e_1 关系符号 e_2〉THEN 〈语句标号〉

(2) IF 〈e_1 关系符号 e_2〉GOTO 〈语句标号〉

(3) IF 〈e_1 关系符号 e_2〉GOSUB 〈语句标号〉

(4) IF 〈e_1 关系符号 e_2〉THEN 〈语句〉

其中：e_1，e_2均为表达式，它可以是数值型的，也可以是字符串型的。e_1关系符号e_2构成了一个关系表达式，它取真（假）值。THEN，GOTO，GOSUB为句中定义符。语句标号是指程序中某个语句的标号。语句可以是除去DEF，REM，END和FOR-NEXT之外的任何一个合法的BASIC语句，当然它也可以是另一个条件语句。

功能：

（1）当程序执行到IF语句时，系统首先对关系表达式进行判断，如果表达式为真（确实成立），则执行其后的附句，即按标号转移或直接执行相应的语句，否则执行IF语句的下一行的语句，从而实现了程序的分支。

（2）前两种句型是等价的，对于第四种句型，只要附

句中不是GOTO语句或另一个取真值的IF语句，则执行完该语句后仍按顺序往下执行。

2.数值型条件语句程序范例

【例12】有一批数目不详的票证，试编写一个程序累加统计其票面数值，并以输入零为标志，结束统计工作。

设以S记累加和，以X记当前输入的票面值，可给出以下流程图（图7－1）和程序。

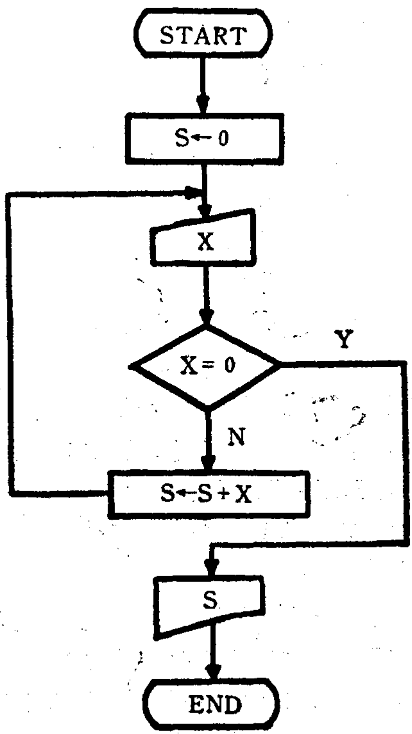


图 7-1

```
10  REM  ADDITIVE  PROGRAM
30  LET  S=0
40  INPUT  "INPUT  X:";X
50  IF  X=0  THEN  GOTO  80
60  LET  S=S+X
70  GOTO  40
80  PRINT
90  PRINT  "S=";S
99  END
```

在正常的累加过程中，每次对40号语句的提问应以非零数值回答，这时50行语句中的关系表达式取假值（即不成立），由60行语句执行累加后返回40行语句，输入下一个数据。当全部票证处理完毕后，对于接下去的提问可用零回答，这时候50行语句的关系表达式取真值（即成立），则转去打印累加结果，并终止程序的执行。显然，50行条件语句使程序的执行产生了两种可能性，即可能执行60行语句，也可能执行80行语句，二者仅居其一。这就是二重分支的例子。如果在一个程序中使用多个条件语句，则可以实现多重分支。下边是一个多重分支的例子：

【例13】编写一个程序，对处于〔0，6〕区间上的 x 值计算函数

$$y=\begin{cases} x-4.5 & (0\leqq x\leqq 2) \\ e^{x}+e^{2x} & (2< x\leqq 4) \\ x^{2}-2x+1 & (4< x\leqq 6) \end{cases}$$

的值。

由于对不同区间上的 x 要用不同的表达式计算，这就要

对所给的 x 值分段加以判断。假定判断是由小到大进行，我们可给出流程图，如图 7－2 所示。

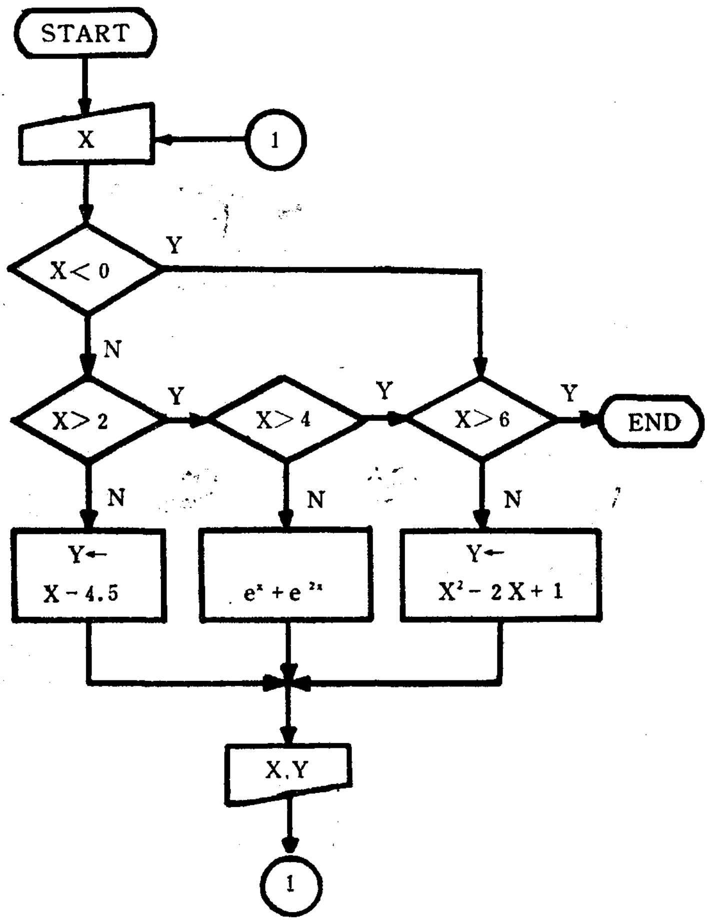


图 7-2

按流程图的逻辑关系很容易写出计算程序。

```
100  REM  MULTIPLE  BRANCH  EXAMPLE
105  INPUT  X
110  PRINT
115  IF  X<0  THEN  GOTO  199
120  IF  X>2  THEN  GOTO  140
125  LET  Y=X-4.5
130  PRINT  "X", "Y1"
135  GOTO  175
140  IF  X>4  THEN  GOTO  160
145  LET  Y=EXP(X)+EXP(2*X)
150  PRINT  "X", "Y2"
155  GOTO  175
160  IF  X>6  THEN  GOTO  199
165  LET  Y=X*X-2*X+1
170  PRINT  "X", "Y3"
175  PRINT  X, Y
180  GOTO  105
199  END
```

这是一个典型的多重分支程序，对于每一个给定的X值，只要执行程序中的一部分语句就可以达到目的。这也是任何多重分支程序在运行时的特点。

条件转移语句的作用固然是实现程序的分支，但从以上两个程序可以看出，它在无条件转移语句的配合之下，也可以有效地完成必要的重复计算任务。从这个意义上来说，它也具有循环语句的功能。那末，当我们对需要重复计算的问题进行程序设计时，何时应采用循环语句，何时应采用转移

语句呢？这应视具体情况而定。一般说来，当需要重复的次数为已知时，用循环语句比较方便；当重复次数不能事先知道，而要在程序的运行中按照问题是否满足一定的条件来决定是否需要继续重复计算时，则应使用条件转移语句配合适当的无条件转移语句来实现。这种情况也是经常遇到的。

3．字符串型条件语句程序范例

在给出具体的例子之前，先介绍一下字符串的大小。它和数值变量类似，其大小表现为两个方面，一是字符串的长度，一是串中每个字符的具体内容。对此，BASIC系统提供了以下四项比较原则：

（1）两个串比较大小的方法是从左到右逐个字符进行比较；

（2）当较短串中的每个字符与较长串中的每个对应字符均相同，则长者为大；

（3）第一个不同字符的ASCII码大的字符串为大，而不问其长度。每个字符的ASCII码请参看附录九；

（4）长度与每个对应字符均相同时，两字符串相等。

例如："ABXYC">"ABXY"

"A">"1234A"

"C">"BASIC"

"TEXT"="TEXT"

【例14】设数区中存放有N种商品，试把C类商品的名称和数量打印出来。

假定每种商品均包括类别、名称、数量三种信息，则可由如下程序来实现。

```
5  REM  STRING  COMPARE  PROGRAM
10 DIM  A$(12), B$(1), X$(4)
```

```
20  INPUT  N
25  PRINT
30  FOR  I=1  TO  N
40  READ  B$, A$, X$
50  IF  B$ <  >"C"  THEN  GOTO  70
60  PRINT  A$, X$, B$
70  NEXT  I
80  DATA  "C", "PEN", "100","A", BIKE",
    "200"
90  DATA  "B", "CAP","1500","C","MAP",
    "880"
100  DATA  "A", "WATCH", "100", "B",
     "SHOE", "2000"
199  END
RUN
? 6
PEN        100        C
MAP        880        C
```

从本例可以看出，比较字符串大小的功能有助于实现情报的检索。程序中的50行语句就是通过比较两字符串是否相等来实现对C类商品的检索的。

【例15】在一批商品中，把数量不超过1000的商品的名称和数量从数据区中检索并打印出来。

如果每种商品仍含有上例中的三种信息，再假定以“•”号作为数区中数据的结束标志，我们可写出如下程序。

```
5   REM  SEARCH  EXAMPLE
10  DIM  A$(1),B$(10),X$(5)
```

```
20  READ  A$
30  IF  A$="*"  THEN  GOTO  199
40  READ  B$, X$
50  IF  LEN(X$)>3  THEN  GOTO  20
60  PRINT  B$, X$
70  GOTO  20
80  DATA  "C", PEN", "100", "A", BIKE",
    "200"
90  DATA  "B", "CAP","1500","C","MAP",
    "880"
100  DATA  "A", "WATCH", "100", "B",
     "SHOE", "2000", "*"
199  END
RUN
PEN       100
BIKE      200
MAP       880
WATCH     100
```

读者可自行分析以上两个程序。在此应当指出：由于在程序中使用了串常数和串变量，有效地扩大了BASIC语言的应用范围，使得它不仅能用于数值计算，而且也能用于各种管理业务中的非数值处理。

以下介绍调用子程序与返回主程序语句。

迄今为止，我们已学过了不少程序语句，也阅读和编写了一些较短的程序。只要稍加留意便可知道，一个程序总是由若干个完成不同功能的程序段组成的。在运行一个程序的时候，这些程序段有的是按顺序执行的，有的是不按顺序执

行的。对于后者，以往我们主要是利用转移（无条件的和有条件的）语句和开关语句，来达到对某些程序段选择使用的目的。然而，由于它的局限性，某些情况下仅用转移和开关语句是很不方便的。例如，要想在一个程序的不同地方多次使用同一个程序段，并要求用完之后返回原处，用转移语句来实现是十分麻烦的。另外，对于一个规模较大的程序，往往需要分段设计，然后联合起来使用，这就要解决程序段之间的联接方式问题，如此等等。

为适应这些需要，方便程序设计，BASIC系统引入了子程序的概念，并设置了有关使用子程序的语句。

今后，我们可以把在程序中使用（允许多次重复使用）的一个相对独立的程序段视为一个子程序，把调用程序视为主程序。下面介绍有关子程序的使用规则。

二十二、调用子程序语句

格式：GOSUB‹语句行号›

其中，句首定义符GOSUB是GO SUBROUTINE的缩写，即调用子程序的意思。语句行号为子程序中的一个语句行号，而且往往是子程序的第一个语句行号。此外，GOSUB语句也可以作为附句出现在条件语句和开关语句中。

功能：

（1）当主程序执行到GOSUB语句时，就按语句行号的指示转去执行子程序。

（2）子程序还可以调用另一个子程序，这称为子程序的嵌套（见图7-3）。但嵌套不能超过24层。

二十三、返回主程序语句

格式：RETURN

功能：

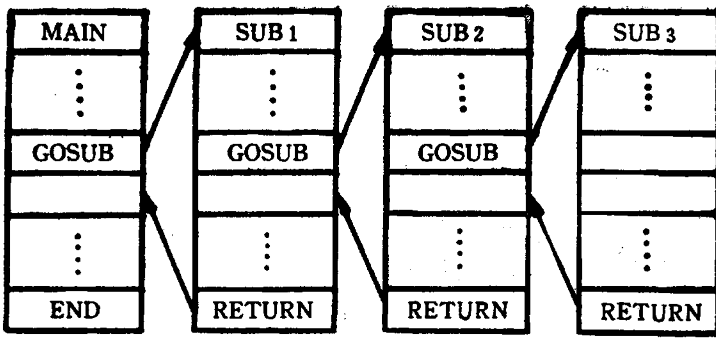


图 7－3　子程序嵌套示意图

（1）当子程序执行到RETURN语句时，则自动返回到主程序中调用该子程序的GOSUB语句的紧接着的下一行语句，继续执行主程序。

（2）一个子程序中最少应有一个RETURN语句作为子程序的出口，而且，在一个完整的程序中（指既有主程序又有子程序的程序），GOSUB 和 RETURN 语句必须同时存在。

程序示例

【例16】计算并打印函数$y=2e^{3x}+4$　在x＝5，6，7处的值，并求其平均值。

这是一个很简单的问题，当然可以用许多种方法编写其计算程序。下面我们试采用子程序的方法编制计算程序，以便分析使用子程序的一些共同性问题。

如果把所给函数的求值过程视为一个独立的部分，编成一个子程序，当需要求某点上的值时，就调用它一次，那么

可给出如图 7－4 所示流程图和程序。

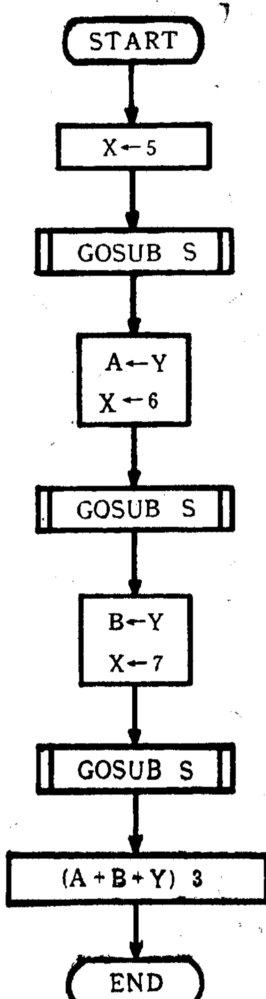


图 7－4

```
10  REM  MAIN  PROGRAM
15  PRINT  "X","Y"
20  LET  X= 5
25  GOSUB  500
30  LET  A=Y
35  LET  X=6
40  GOSUB  500
45  LET  B=Y
50  LET  X= 7
55  GOSUB  500
60  LET  P= (A+B+Y)/3
65  PRINT  "P="; P
99  END
500  REM  THE  SUBROUTINE
510  LET  Y=3*X
520  LET  Y=2*  EXP(Y)
530  LET  Y=SQR(Y+ 4 )
540  PRINT  X,Y
550  RETURN
RUN
X        Y
   5          2556.96
   6          11459.5
   7          51357.7
P=21791.4
```

对这个程序，可作以下两点说明：

（1）10～99号语句为主程序部分，500～550号语 句 为

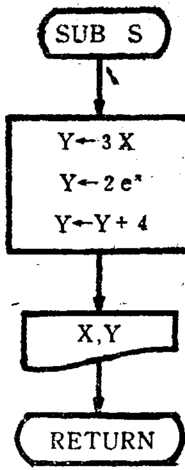


图 7－5

子程序部分。主程序曾三次调用子程序。每次调用之前，先为运行子程序所需要的有关变量提供信息（即给有关变量赋值）。该程序是通过20，35，50三个语句给变量X赋值的，这些值在每次调用时分别传送给子程序。每当子程序运行完一遍而从RETURN语句返回时，就自动地把结果信息（在这里就是求出的Y值）带回主程序。三次调用分别返回到30，45和60号语句，然后再按照问题的要求作适当的处理。

（2）在BASIC中，主程序和子程序所用的变量具有同一性。主、子程序之间的信息就是靠这种同一性来传递的。需要指出的是，虽然子程序在逻辑上是一个完整的程序段，但由于一般说来它又是通用的，正是由于这种通用性，子程序具体执行完全依赖于主程序的安排。

【例17】求从J个元素中每次取K个元素的组合数

$$C_J^K = \frac{J!}{K!\ (J-K)!}$$

显而易见，本题的主要计算工作是求阶乘，因此我们可以编写一个求正整数阶乘的子程序，供计算组合数时调用。下边直接给出这个程序。

```
10  REM  COMBINTIAN  MAIN  PROGRAM
20  PRINT  "INPUT  VALUES  OF  J,K"
30  INPUT  J,K
40  LET  N=J
50  GOSUB  300
60  LET  N=K
70  LET  C=F
80  GOSUB  300
90  LET  C=C/F
100  LET  N=J-K
110  GOSUB  300
120  PRINT
130  PRINT  "COMB=";C/F
199  END
300  REM  FACTORIAL  SUBROUTINE
     OF  N
310  LET  F=1
320  FOR  I=1 TO  N
330  LET  F=F*I
340  NEXT  I
350  RETURN
```

```
RUN
INPUT  VALUES  OF  J,K
? 10    ? 3

COMB=120
```

通过上述两例，我们可以概括出子程序的程序设计要领。这就是：在主程序中应恰当安排调用点，作好调用准备以及调用返回后的必要处理；对于子程序则应充分注意到其逻辑上的完整性和程序的通用性。按照这些要领，读者仔细分析一下上述程序将是有益的。

【例18】设某类电报的计费办法是：普通电报30字以内计费5元，超过30字时，每多一个字增加0.2元；甲级电报费为普通电报费的两倍。试编写一个计算电报费的程序。

从题意可知，无论什么电报均要计算普通电报费，所以可把普通电报费的计算写成一个子程序。为了作好调用准备和返回处理，我们约定：若为甲级电报时，则以负字数输入。据此分析，可给出如图7-6所示的流程图和下列程序。

```
10  REM  TELEGRAM  FEE MAIN  PROGRAM
20  PRINT  "INPUT NUMBER OFF WORDS N"
30  INPUT  N
40  PRINT
50  IF  N>0  THEN  GOTO  100
60  LET  N=-N
70  GOSUB  400
80  LET  A=2*A
90  GOTO  110
100  GOSUB  400
```

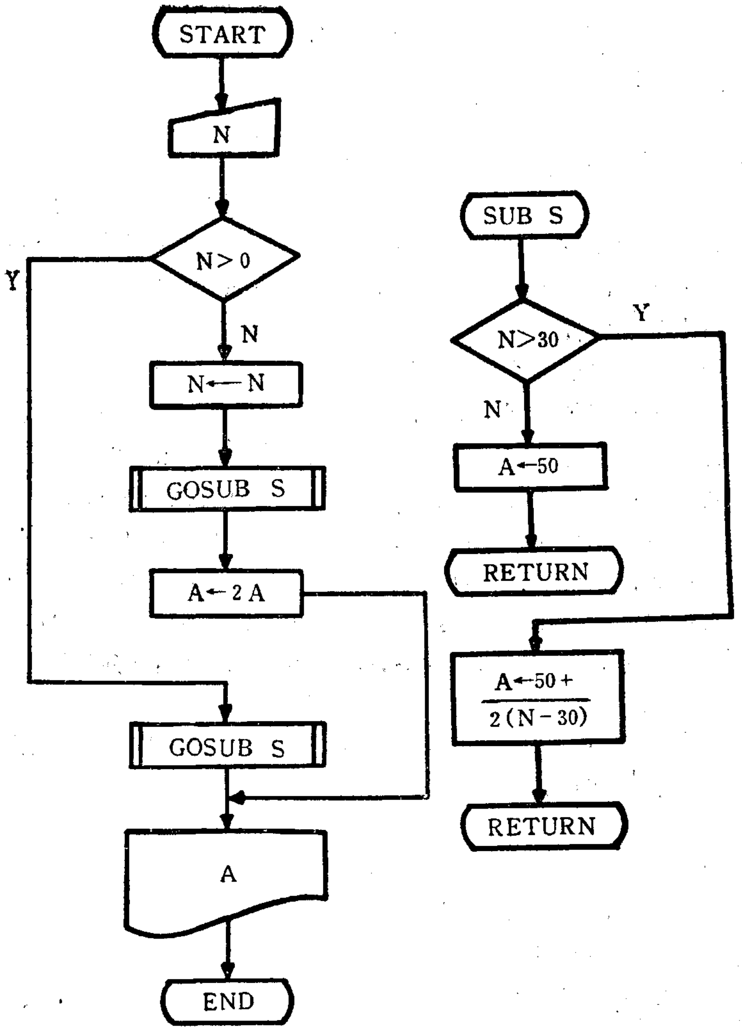

START
N
N>0
Y
N
N←−N
GOSUB S
A←2A
GOSUB S
A
END
SUB S
N>30
Y
N
A←50
RETURN
A←50+
2(N−30)
RETURN

图 7-6

```
110 PRINT "TELEGRAM FEE=";A
199 END
400 REM SUBROUTINE
410 IF N>30 THEN GOTO 440
420 LET A=50
430 RETURN
440 LET A=(N-30)*2+50
450 RETURN
RUN
INPUT NUMBER OF WORDS N
  ? 40
TELEGRAM FEE=70
```

这是一个在子程序中安排了两个出口的例子，按照不同情况于不同地方退出子程序而返回主程序。上述程序是根据字数大于30时进行程序分支的，分支上分别设置出口实现返回。由此可见，当子程序需要进行分支处理时，设置多个出口是方便的。

【例19】把N个数中的负数选出来，将其放入一个一维数组里并打印之。

若把存放负数于数组里的过程视为相对独立的部分，则可写出如下的程序。

```
5  REM SELECT AND PRINT NEGATIVE
   NUMBERS
10 INPUT N
15 PRINT
20 DIM M(N)
25 LET Y=0
```

```
30  FOR  I=1  TO  N
35  READ  X
40  IF  X  <  0  THEN  GOSUB  500
45  NEXT  I
50  FOR  K=1  TO  Y
55  PRINT  M(K),
60  NEXT  K
65  DATA  1, -1.5, 2, -2.5, 3, -3.5, 4,
     -4.5, 5, -5.5
99  END
500  LET  Y=Y+1
510  LET  M(Y)=X
520  RETURN
RUN
?  10
-1.5     -2.5     -3.5     -4.5     -5.5
```

这个程序使用IF—GOSUB句型，调用子程序语句是作为一个附句出现的。应注意的是，它的返回点仍为调用语句的下一个语句，在本例中就是45号语句。读者可根据本例自己试设计一个流程图。

二十四、开关语句

开关语句也叫做控制转向语句。这类语句不仅在某些情况下可以大大简化用条件语句实现程序分支的手续，而且它能有力地实现BASIC程序本身的自动控制功能。从而有利于程序的编制和提高程序运行的效率。BASIC中的开关语句共有三类，以下分别给予介绍。

1．第一类开关语句

格式：ON〈数值表达式〉 THEN〈n_1，n_2，…，n_k〉

ON〈数值表达式〉 GOTO〈n_1，n_2，…，n_k〉

ON〈数值表达式〉GOSUB〈n_1，n_2，…，n_k〉

其中：n_i为本程序的语句标号。每个标号隐含一个顺序号，这些顺序号从左到右依次为1，2，3，…，k。

功能：

当程序执行到这类开关语句时，系统首先计算数值表达式的值并取整数。一般说来，该整数应等于其后某一个语句标号的顺序号。然后程序自动转去执行由此顺序号决定的那个标号的语句，从而起到接通某路开关的作用。如果所得的整数小于1或大于最后一个顺序号，则此开关语句不起作用（实际上也可以视为接通了一路开关，程序顺序往下执行。

【例20】求一元二次方程

$$ax^2+bx+c=0 \quad (a\neq 0)$$

的根。

因为这类方程有统一的求根公式，根的类型又取决于判别式的符号，所以可用判别式构造开关语句的数值表达式，对不同类型的根分别加以处理。下面给出它的流程图（见图7-7）和程序。

```
10  REM  ROOTS  OF  A*X*X+B*X+C=0
20  DIM  C$(7), E$(5), R$(4)
30  READ  C$, E$, R$
40  PRINT  "INPUT  A, B, C"
50  INPUT  A, B, C
55  IF  A=0  THEN  GOTO  210
60  PRINT
70  LET  R=-B/(2*A)
```

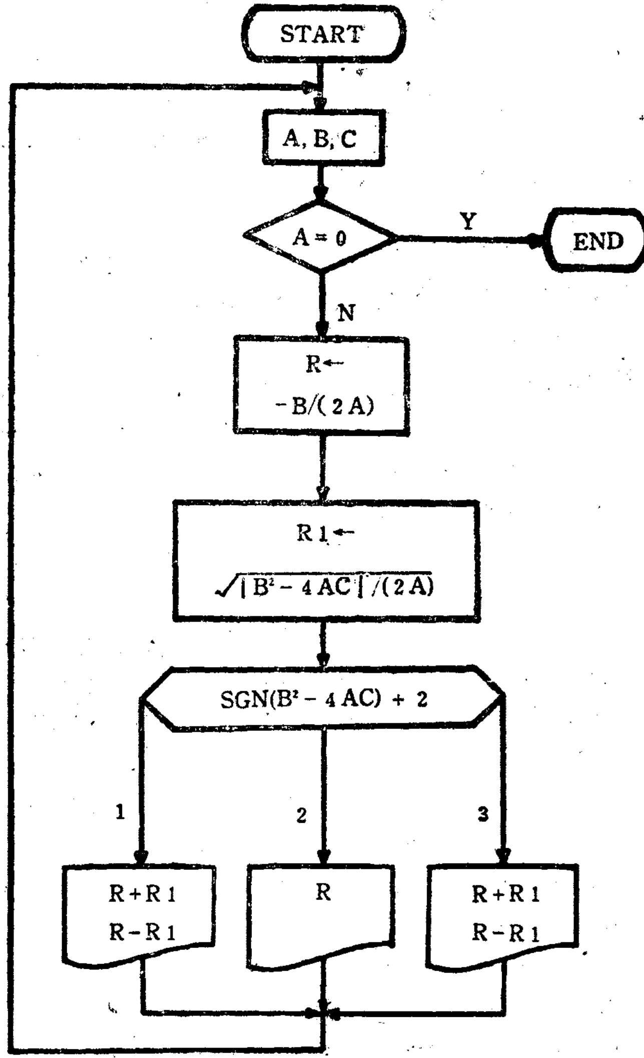

START
A, B, C
A = 0
Y
END
N
R←
-B/(2A)
R1←
√|B² - 4AC| /(2A)
SGN(B² - 4AC) + 2
1
2
3
R+R1
R-R1
R
R+R1
R-R1

图 7-7

```
80  LET  R1=SQR(ABS(B*B-4*A*C))
    /(2*A)
90  ON  SGN(B*B-4*A*C)+2  GOTO  100,
    140, 170
100  PRINT  C$
110  PRINT  "X1=";  R;  "+";  R1;  "I"
120  PRINT  "×2=";  R;  "-";  R1;  "I"
130  GOTO  50
140  PRINT  E$
150  PRINT  "X1=X2=";  R
160  GOTO  50
170  PRINT  R$
180  PRINT  "X1=";  R+R1,  "X2=";  R-R1
190  GOTO  50
200  DATA  "COMPLEX", "EQUAL", "REAL"
210  END
```

通过运行以上程序，读者将体会到，这类开关语句的关键在于：按照问题的特点恰当地构造一个数值表达式，以期实现开关的功能。这类开关语句，适用于一切可借助于自然数序列进行控制的问题。

2．第二类开关语句

格式：ONERRGOTO‹行号›

其中，ONERR表示程序发生语法（句法）错误的意思，语句行句表示出错开关接口。

功能：如果在程序中有此类语句（此类语句可设在程序中的任意位置），那么当系统在程序运行过程中检测出语法错误时，程序就自动转向出错开关接口。

应当指出的是，这一功能绝不只是单纯用于出错处理的消极措施，这一功能还具有更积极的意义。例如，可以在程序中人为地设置某种语法错误，利用系统的语句检测功能，达到程序自身进行控制的目的。

格式：RESUME

功能：RESUME语句与ONERR语句搭配使用，含意是重新再做一次。当处理错误的程序段结束时，用来使程序返回当初发生错误的语句处，并使程序继续按顺序运行。

【例21】打印出N个数中非负数的算术平方根。

程序为：

```
10  REM  PRINT  SQUARE  ROOT
15  INPUT  N
20  ONERR  GOTO  40
25  I=I+1
26  IF  I=N  THEN  50
30  READ  X
35  PRINT  "SQR("X")=";   SQR(X),
40  GOTO  25
45  DATA  4, -9, 16, -25, 36, -449, 64,
     -18, 100, 190
50  END
```

当以上程序遇到负数开平方时，可通过出错开关接口实现自动转移。

【例22】用于处理输入错误的程序。

```
10  ONERR  GOTO  80
20  INPUT  "X=";  X
30  PRINT  1/X
```

```
40  ONERR  GOTO  110
50  INPUT  "Y=";  Y
60  PRINT  SQR(Y)
70  STOP
80  PRINT  "DATA  MUST  NOT  BE  ZERO"
90  INPUT  "X=";  X
100  RESUME
110  PRINT  "DATA  LESS  THAN  ZERO"
120  INPUT  "Y=";  Y
130  RESUME
```

请读者自行分析，并运行该程序。试分别输入X=5，0，-5及Y=2，8，-2，-8，0等，观察程序运行的状况。

二十五、自定义函数语句

我们已经知道，FP BASIC 语言有自己的系统函数。系统函数也叫做标准函数。在程序设计中，当用到标准函数时，只要按规定正确地引用它们就可以了。但是，一方面由于标准函数的数量有限，另一方面有时为了在程序设计中避免多次重复书写同一个复杂的表达式，BASIC系统向用户提供了自定义函数语句，允许程序设计人员自己定义所需要的函数并使用它们。下面介绍自定义方式和用法。

1．自定义函数的格式和功能

格式：DEF FN A(X)=数值表达式

其中：DEF——语句定义符，DEF是DEFINE的缩写；

FN——自定义函数标记，FN是FUNCTION的缩写；

A——自定义函数名，它可以是A～Z中的任一字母；

X——形式参量，它可以是任何一个简单变量。

数值表达式包括形式参量的任何合法的BASIC数值表

达式。

功能：

（1）该语句把等号右边的表达式视为一种函数，而用左边的等价记号来代表它。凡是在程序中用到这个表达式的地方，都可以用等价记号来引用它。

（2）在引用自定义函数时，形式参量必须以实在参量来代替。所谓实在参量，指的是数值常数、已赋值的数值变量以及对变量已赋值的数值表达式。

（3）从对语句的格式规定可知，系统允许用户在一个程序中使用26种自定义函数。

（4）在引用时，一个自定义函数可以嵌套另一个自定义函数，但不能嵌套自身，而且嵌套层次不可多于八层。此外，还应注意不要使嵌套后的表达式太复杂，以免引起错误。

2. 程序示例

【例23】计算并打印函数

$$\sin^2 x + e^{3\sin x+2} + \ln^2 x + e^{3\ln x+2}$$

分别在$x = 0.5$，1.6，2.7，3.8处的值。

我们直接写出如下程序：

```
10  REM  FUNCTION  CALCULATION
15  ONERR  GOTO  99
20  PRINT  "X", "Y"
25  DEF  FN  T(X)=X*X+EXP(3*X+2)
30  READ  X
35  LET  Y=SIN(X)
40  LET  Z=LOG(X)
45  PRINT  X, FN  T(Y)+FN  T(Z)
50  GOTO  30
```

```
55  DATA  0.5, 1.6, 2.7, 3.8
99  END
RUN
X        Y
 .5        82.7671
1.6        179.709
2.7        173.242
3.8        408.789
```

自定义函数本质的特点，在于能从总体上抽象地描述函数的一般结构。这主要体现在形式参量的选择上。读者应从25号语句定义的表达式与题目所给函数的对照中找出这种特点。至于函数的具体形式，则完全取决于在引用自定义函数时用何种实在参量去代替形式参量。本程序的35和40号语句，给出形式参量X提供的实在参量。

【例24】在区间〔1，2〕上以0.1为步长，打印双曲线函数

$$\cosh x=(e^{x}+e^{-x})/2$$

$$\sinh x=(e^{x}-e^{-x})/2$$

的值。

```
10  REM  PRINT  HYPERBOLIC  FUNCTION
15  DEF  FN  C(X)=(EXP(X)+EXP(-X))/2
20  DEF  FN  S(X)=(EXP(X)-EXP(-X))/2
25  PRINT  "HYPERBOLIC  FUNCTION"
30  PRINT  "X", "COSH(X)", "SINH(X)"
35  FOR  Y=1  TO  2  STEP  0.1
40  PRINT  Y, FN  C(Y), FN  S(Y)
45  NEXT  Y
```

```
99  END
RUN
HYPERBOLIC  FUNCTION
X        COSH(X)        SINH(X)
1            1.54308        1.1752
1.1          1.66852        1.33565
1.2          1.81066        1.50946
1.3          1.97091        1.69838
1.4          2.1509         1.9043
1.5          2.35241        2.12928
1.6          2.57747        2.37557
1.7          2.82832        2.64564
1.8          3.10748        2.94218
1.9          3.41774        3.26817
```

从以上两个例题可以看出，一旦在程序中把某一个表达式定义为自定义函数，它就能象标准函数一样，可以在程序中随意地引用，这对于处理工程技术中一些复杂的表达式是十分方便的。

【例25】求复合函数

$$y=f(g(x))$$

在$x=1$，9处的值。(a，b，c的值任意指定)。

其中：

$$f(g)=\sin^2 g+\cos g$$

$$g(x)=ax^2+bx+c$$

程序为：

```
10  REM  COMPOUND  FUNCTION
15  ONERR  GOTO  99
```

```
20 DEF  FN  F(S)=SIN(S)*SIN(S)+COS(S)
25 DEF  FN  G(T)=A*T*T+B*T+C
30 READ  A, B, C
35 READ  X
40 PRINT  X, FN  F(FN  G(X))
45 GOTO  30
50 DATA  2, 3, -1, 1, 3, 5, 7, 9
99 END
RUN
1        -8.08932E-2
9        1.0453
```

在使用自定义函数时，应注意以下两点：

1）在规定范围内，函数名称的选取完全是任意的；但一旦选定，就应在程序中始终使用该名称，不可随意改动；

2）在规定范围内，形式参量的选取也是完全任意的，但等式两边所用的参量必须一致。

读者可试改变上例的函数名称和形式参量，再运行程序，观察所得结果。

最后，应该指出：对于许多算法来说，使用自定义函数，可使算法具有通用性。

习题与思考题

1. IF 语句有哪些功能，其主要功能是什么？

2. 利用循环语句和转移语句均可实现重复性计算，在程序设计中如何选择使用它们？

3. 用条件语句编写一个程序，求一元二次方程的实根。

$$ax^2+bx+c=0 \quad (a\neq 0)$$

4．设1981年我国人口约为10亿，按年增长率分别为千分之二十，千分之十和千分之一进行计算，到2001年时，若人口超过12亿则打印标记 YES，否则打印标记 NO。可采用以下计算总体量的公式：

$$P=P_0(1+R\%)^N$$

5．某类货物的运费规定以50公斤为界限，按如下办法计算。

$$F=\begin{cases} ax & (x\leqq 50) \\ 50a+(x-50)b & (x>50) \end{cases}$$

（其中 x 为货物重量，a 和 b 为每公斤货物的运费，且 $b>a$），试编写一个计算运费的程序。

6．试编写一个程序，统计一批数中零，正数，负数的个数。

7．试编写一个程序，求一批非零数中正数的自然对数和负数的绝对值的平方根。

8．编写一个程序，统计考试成绩在60分以下，60～80分以及81～100分各范围内的考生人数，并以－1为标记，作为程序的运行的终止。

9．在 BASIC 程序设计中，使用串常数和串变量有何重要意义？

10．编写一个拼写字符串的程序。

已知：A＄＝"OPEN　FILE"

　　　B＄＝"BASIC　ENTER　RDOS　SYSTEM"

求：C＄＝"ENTER　RDOS　BASIC　FILE"

11．编写一个程序，把N个职工的姓名、性别、工龄以

串常数的形式放在数区中，并把10年以上（包括10年）工龄职工的有关信息打印出来。

12．编写一个程序，实现把二维数组$A(m, n)$各元素的值以行的顺序传送到一个一维数组中（先绘出流程图）。

13．主程序和子程序之间的信息是怎样传递的，在设计这类程序时应注意那些要点？

14．用 RND (X)函数产生三组（每组20个）随机数，用调用子程序的方法求出每组的最大者并打印出来。

15．设渐开线的参数方程为

$$x=2(\cos t+t\sin t)$$

$$y=2(\sin t-t\cos t)$$

试编写一个程序，对于$0\leqq t\leqq 6$，步长为0.1，打印出x，y的相应值。

16．设摆线的参数方程是

$$x=t-\sin t$$

$$y=1-\cos t$$

试编写一个程序，对于$0<x<1$，步长为0.1，打印出x，y的相应值。

17．设计一个程序，计算5个商店，10种商品每个月（30天）每种商品的零售总金额及每个商店10种商品的零售总金额。

18．编写一个程序，计算两个方阵（10×10）的乘积，并打印结果。

19．设计一个打印菱形图案的程序。

20．仔细领会自定义函数的功能及本质的涵义。

21．用自定义函数语句编写一个程序，计算函数

$$w=\sin^3 x+\sin x+\ln^3 x+\ln x+10$$

在 $x=0.5$处的值。

22．用自定义函数语句编写一个程序，计算五次多项式

$$a_5x^5+a_4x^4+a_3x^3+a_2x^2+a_1x+a_0$$

在区间〔0，1〕上的值（步长取0.05）。

23．分别用一般方法和自定义函数方法编写程序，计算函数

$$R=\frac{e^{2x}+1+2e+3e^{2x}}{15.3+\sin^2x+1+2\sin x+3\sin^2x}$$

的值。

24．选一个复合函数，利用〔例48〕的程序进行计算。

Retro Workshop 果粉工作室

第八章　FP BASIC 的图形、发音功能及其程序设计方法

本章主要介绍FP　BASIC 的图形、发音功能及其程序设计方法。FP　BASIC 除了有正文显示方式外，还有低分辨率图形及高分辨率图形这两种显示方式。

第一节　低分辨率图形

低分辨率图形有如下几种语句。

一、设置低分辨率图形语句

格式：GR

功能：将屏幕设定为低分辨率图形显示方式，即屏幕上部为40×40的图形区域，下部留 4 行作显示程序正文之用，这种显示方式称为图文混合（GR）方式。

若要由图文混合方式转入全屏显示方式（不留下部 4 行作显示程序文本用），则使用置全屏低分辨图形指令：POKE-16302，0。

若要由全屏显示方式回到图文混合方式，则用返回图文混合型指令：POKE-16301，0。

二、选择低分辨颜色语句

格式：COLOR＝<*n*>

其中*n*从 0 到15（共16种颜色）中取值。序号与颜色的对应关系列于下表中：

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
黑	深红	深蓝	紫	深绿	灰	蓝	浅蓝	橘	橙	灰绿	粉红	绿	黄	浅绿	白

三、定位画点语句

格式：PLOY‹*X*›，‹*Y*›

功能：在水平坐标为*X*及垂直坐标为*Y*处绘出一个色点。*X*及*Y*从0～39的范围内取值。坐标原点（0，0）在屏幕的左上角。

四、绘水平线语句

格式：HLIN‹X1›，‹X2›AT‹Y›

功能：从点(X1，Y)到点(X2，Y)绘出一条水平线。

五、绘垂直线语句

格式：VLIN‹Y1›，‹Y2›AT‹X›

功能：从点(X，Y1)到点(X，Y2)绘出一条垂直线。

六、定点查色值语句

格式：SCRN(X，Y)

功能：取得屏幕上点(X，Y)的颜色。

以下给出相应的范例，供读者练习。

【例1】画出一方框。

程序为：

```
5   GR：COLOR=12：HOME
10  HLIN  0，39  AT  0
15  VLIN  19，0  AT  39
20  HLIN  39，0  AT  19
25  VLIN  19，0  AT  0
30  FOR  I=0  TO  400：  NEXT  I
```

```
40  DEL  10, 10
45  GOTO  5
```

试运行此程序，观察显示的结果。

【例 2】请用以下简短的程序，画一张漂亮的“地毯”。

```
10  GR:POKE  49234, 0
20  FOR  X=0  TO  39
30  COLOR=INT(RND(1)*16)
40  HLIN  X, 39-X  AT  X
50  VLIN  X, 47-X  AT  39-X
60  HLIN  39-X, X  AT  47-X
70  VLIN  47-X, X  AT  X
80  NEXT  X
```

其中POKE 49234，0为置全屏于低分辨率图形显示方式。

运行以上程序，将看到屏上显示出各种颜色的低分辨率图形，构成一幅漂亮的“地毯”图案！

【例 3】下面给出一个在分区数$N \leqslant 14$情况下绘制直方图的通用程序。

```
8   GR:COLOR=3
10  INPUT"N=";N
20  DIM  D(N)
30  S=0:FOR  I=1  TO  N
40  READ  D(I):S=S+D(I)
50  NEXT  I
60  K=INT(28/N)
65  HOME
```

```
70  FOR  I=1  TO  N
80  FOR  J=3+(I-1)*K  TO  7+I*K
90  IF␣N>8  THEN  110
100 VLIN  39-D(I)*60/S, 39  AT  J: GOTO  120
110 VLIN  39-D(I)*80/S, 39  AT  J
120 NEXT  J
130 NEXT  I
140 END
150 DATA  34, 15, 23.5, 9, 11.3
```

请上机执行此程序，并理解之。

【例 4】用全屏作低分辨图形（夜晚闪星程序）。

```
90  REM  STARRY  NIGHT  PROGRAM
95  DARK=60
100 GR
110 REM  SET  TO  FULL-SCREEN  GRAPHIC
120  POKE  -16302, 0
130 REM  CLEAR  BOTTOM  OF  SCREEN
140 COLOR=0:  FOR  I=40  TO  47:  HLIN
    0, 39  AT  I:  NEXT  I
150 REM  PLOT  STARS  AND  SPACES
160 COLOR=15*RND(1)
170  PLOT  40*RND(1), 47*RND  (1)
180  COLOR=0
190  FOR  I=1  TO  DARK
200  PLOT  40*RND(1), 47*RND(1)
210  NEXT  I
220  REM  HAS  ANY  KEY  BEEN  PRESSED?
```

（注意已按下了什么键？）

```
230  IF  PEEK(-16384)<127  THEN  GOTO
     160
240  POKE  -16368,0
250  DARK=DARK-30:  IF  DARK<0
     THEN  DARK=60
260  GOTO  160
```

在-16384位置上的数值，通常是小于127的，当触动到任何键（除RESET、CTRL、REPT或SHIFT 键外）时，此数值就突然增大到超过127。因而，通过每次穿越循环来测试这个位置，就能近似地说出什么时候已按动过一个键，并且使该程序相应地执行某指令。在读键盘的PEEK指令后，跟随一个POKE-16368，0语句，这是一个好的格式，它使键盘复位，以便使程序能看出下一次有键被触动。在每次触动任何一个键（按空格键条是特别方便的）时，“天空”将发亮，等待一会，看看效果，然后再次触动键，DARK的值将为零，而“天空”全部亮了——早晨好，最后可以再按一个键，“夜晚闪星的天空”就出现了。

试打入RUN命令，运行本程序。

【例5】一个低分辨绘图程序。

```
NEW
10  GR
20  FOR  ROW=0  TO  30  STEP  10
30  FOR  COLUMN=0  TO  35  STEP  5
40  COLOR=RND  (1)  *16
50  FOR  I=0  TO  9
60  HLIN  COLUMN, COLUMN+4  AT  ROW
```

```
   +1
70  NEXT  I
80  FOR  DE=1  TO  100
90  NEXT  DE
100  NEXT  COLUMN
110  NEXT  ROW
120  GOTO  20
```

这是一个两层嵌套的例子。在进入下一例题之前，先运行此程序。注意，当使用FOR语句写程序时，每一个FOR必须配有NEXT。

【例6】一个错误的程序。

```
NEW
FOR  I=10  TO  20
PRINT  I
FOR  J=30  TO  40
PRINT  J
NEXT  I
NEXT  J
```

这程序不能工作，因为它的循环互相交叉，而提供了没有任何意义的错误的信息。当工程中需要交叉循环时，应使用IF语句构造的循环。

【例7】嵌套循环的一个绘图程序。

```
NEW
10  GR
20  FOR  ROW=0  TO  30  STEP  10
30  FOR  COLUMN=0  TO  35  STEP  5
40  COLOR=RND  (1)  *16
```

```
50  FOR  I=0  TO  9
60  HLIN  COLUMN, COLUMN+4  AT  ROW
      +1
70  NEXT  I
80  FOR  DE=1  TO  100
90  NEXT  DE
100  NEXT  COLUMN
110  NEXT  ROW
120  GOTO  20
```

这程序有三层嵌套，画出的是床单图案。试去掉80和90行，看看会出现什么情况？再试添上一行：

```
55  COLOR=16*RND(1)
```

看看又会出现什么情况？这些情况是你所预期的吗？

【例8】下面是一个使一颜色点(球)以屏幕的左右为边界，横跨屏幕来回跳动的程序。

```
NEW
400  REM  CHOOSE  A  COLOR  FOR  THE
       "BALL"
       (选球的颜色)
420  BALL=9
440  REM  SET  GRAPHICS  MODE
       (置作图方式)
460  GR
480  REM  CHOOSE A STARTING  POSITION
       FOR  THE  BALL
       (选择球的初始位置)
500  XOLD=20
```

```
520  REM MOVE THE BALL BACK AND
     FORTH BY ADDING AN INCRNMENT
     (CALLED X SPEED) TO THE X
     POSITION
     （通过在X位置上加上一递增量——X速度，使
     球前后移动）
540  REM TO MOVE THE BALL TO THE
     LEFT, MAKE THE INCHREMENT
     NEGATIVE
     （要使球往左移动，则使递增量为负）
560  XSP=1
580  REM TO CALCULATE THE NEW
     X POSITION BY ADDING THE IN-
     CREMENT X SPEED TO THE OLD
     X POSITION
     （在原来X位置上加上递增量，标出X新位置）
600  XN=XOLD+XSP
620  REM CHECK THAT THE BALL
     WOULD BE ON THE SCREEN
640  IF (XN>-1) AND (XN<40) THEN
     GOTO 740
660  REM THE BALL WOULD BE OFF
     ONE SIDE OF THE SCREEN, SO
     CHANGE THE DIRECTION OFF X
     MOTION
     （球将离开屏幕的一边，所以改变X运动方向）
680  XSP=-1*XSP
```

```
700  GOTO 600
720  REM PLOT THE BALL IN IT'S
     NEW POSITION
    （在新的位置上画出球）
740  COLOR = BALL
760  PLOT XN, 20
780  REM ERASE THE OLD BALL POSI-
     TION
    （擦除球的原来位置）
800  COLOR = 0
820  PLOT XOLD, 20
840  REM SAVE THE CORRENT BALL
     POSITION FOR NEXT MOVE
    （为一次运动而保存当前球的位置）
860  XOLD = XN
880  REM MOVE AGAIN
900  GOTO 600
```

如果改变"XSP"的值，就很容易理解变量 XSP (X SPEED)为什么叫做这一名字（X速度）。就这一例子，试使X SPEED = 2，运行此程序。如果把X SPEED的值定得太高，则球将很快跳过屏幕，点与点之间没有痕迹。

这类程序是许多典型电视游戏的基础，用些时间运行这一程序是值得的。试改变程序中的某些部分，看看用它能作些什么。

第二节　高分辨率图形

高分辨率图形有如下几种语句。

一、设置高分辨率图文混合型语句

格式：HGR

功能：将屏幕（显示区首页）设定为高分辨图形状态，即屏幕上部为280×160的图形区域，下部留4行作显示程序正文之用。这种状态称为“高分辨图文混合型”（HGR）状态。

若要由HGR状态转入全屏显示（HGR2）状态，则用指令：POKE-16302，0。

若要由HGR2状态返回HGR状态，则用指令：POKE-16301，0。

二、设置全屏高分辨图形语句

格式：HGR2

功能：设定屏幕次页为高分辨图形状态，使屏幕变成280×192的图形区域（不再留底部4行给程序正文用了），并清除全屏（全屏显黑色）。

三、设定绘图颜色语句

格式：HCOLOR =〈n〉

其中$n = 0 \sim 7$，共有6种颜色可选择（因为有两种已互相重迭）。

0	1	2	3	4	5	6	7
黑	绿	紫	白	黑2	橘	蓝	白2

四、定位画点语句

格式：HPLOT〈X〉，〈Y〉

功能：在水平坐标为X及垂直坐标为Y处画出一个色点。X的范围：0～279；Y的范围：0～159（HGR状态）或0～191（HGR2状态）。注：原点（0，0）在屏幕的左上角。

五、绘线段语句

格式：HPLOT〈X1〉，〈Y1〉TO〈X2〉，〈Y2〉

功能：从点（$X1$，$Y1$）到点（$X2$，$Y2$）画出一段高分辨率的直线，命令可以延伸至更多的点（Xn，Yn），以便画出许多连续的线段。

例如：以下一个语句便可画出三段互相连接的线段：

HPLOT 40，50 TO 80，90 TO 130，150 TO 40，50

下面提供几个范例，供读者练习。

【例9】画一艘"飞船"。

程序为：

```
10  REM  USES  HPLOT  TO. DRAW  ELE-
    MENTARY  SHIPS
12  HOME:HGR:HCOLOR=3
20  HPLOT  120,120  TO  120,50  TO  140,20
    TO  160,150  TO  160,120
30  HPLOT  TO  180,170  TO  120,120
40  HPLOT  120,120  TO  120,170  TO  160,170
    TO  160,120
```

【例10】画"曲线"。

程序为：

```
5   REM  STRAIGHT  LINES-CURVE
10  HGR2:HCOLOR=3
```

```
20  FOR  Y=0  TO  191  STEP  4
30  HPLOT  0,Y  TO  279*Y/191,191
40  NEXT  Y
```

其中，第30行画线，曲线的逼近，用的是"线绸"的手法。

【例11】画花纹程序。

```
1   REM  DRAW  OVERLAPPING  RAND
    PATTERNS
5   HGR2:A=5
10  X1=RND(1)*279:Y1=RND(1)*191
20  C=C+1: IF  C=8  THEN  C=0
30  HCOLOR=C:Y=191
40  FOR  X=0  TO  279  STEP  A
50  HPLOT  X,191  TO  X1,Y1  TO  279  -X,0
60  NEXT  X
70  FOR  Y=0  TO  191  STEP  A
80  HPLOT  279,191-Y  TO  X1,Y1  TO  0,Y
90  NEXT  Y
100  FOR  I=1  TO  500:NEXT  I
110  IF  PEEK(-16384)=155  THEN  POKE
     -16368,0:TEXT
120  GOTO  10
```

其中，第10行选定一个随意的图形中心。这个程序连续画出许多个互相重叠的图案，每画一个改变一次颜色(20, 30两行)。第100行用于延时，使程序在画完一个图案时停一停，以便使用者能看清楚图形。

第110行跳出程序，按下ESC键会使程序结束。

读者不妨多欣赏几次这个漂亮的图案。

第三节　发音功能语句

格式：PEEK(-16336)

　　　POKE-16336,0

功能：使计算机中的扬声器发出"哔"的声音。

任何一个重复-16336的PEEK或POKE的程序，都可以形成稳定的音调。见下例。

【例12】产生弹球声音的程序。

下面是一组使弹球发出声音的程序：

```
240 REM SET S TO ADDRESS OF
    SPEAKER
    (置S的内容于扬声器的地址)
250 S=-16336
270 REM MAKE A BOUNCE NOISE
    (产生弹球的声音)
280 FOR I=1 TO 5
290 BOUNCE=PEEK(S)-PEEK(S)+PEEK (S)
    -PEEK(S)
300 NEXT I
310 REM MAKE A BOUNCE NOISE
    (产生一弹球的声音)
320 FOR I=1 TO 5
330 BOUNCE=PEEK(S)-PEEK(S)+PEEK(S)-
    PEEK(S)
340 BOUNCE=PEEK(S)-PEEK(S)+PEEK(S)
    -PEEK(S)
```

```
850  NEXT  I
860  GOTO  280
```

第四节　游戏控制程序语句

格式：PEEK(－16286)
　　　PEEK(－16287)

PEEK(－16286)的功能是说出0号控制器上的按钮是否已被按下；

PEEK(－16287)的功能是说出1号控制器上的按钮是否已被按下。

在用户的系统上若配有游戏控制器，可以试运行以下程序：

```
10  CALL  -936
20  VTAB  10
30  PRINT "BUTTON  ZERO=";PEEK(-16287)
    >127
40  PRINT  "BUTTON  ONE=";PEEK(-16286)
    >127
50  GOTO  10
```

如果相应的按钮被按下，则第30行和第40行中的PEEK将大于127。

第五节　编 程 简 例

绘制一幅动画——飞鸟的程序

1．绘出鸟的造成结构图

首先我们设计一只鸟的造型结构图，如图8-1所示。此图有16个点，由16个X坐标和16个Y坐标，利用HPLOT绘线的方法，可将此图绘制在屏幕上。但绘制时须注意：由于图形的中心为（0，0），而屏幕中心为（140,80），且Y轴向下，因此必须作如下坐标变换才能描点：

（1）$X \leftarrow X+140$；

（2）$Y \leftarrow Y+80$；

（3）Y的方向变为$-Y$。

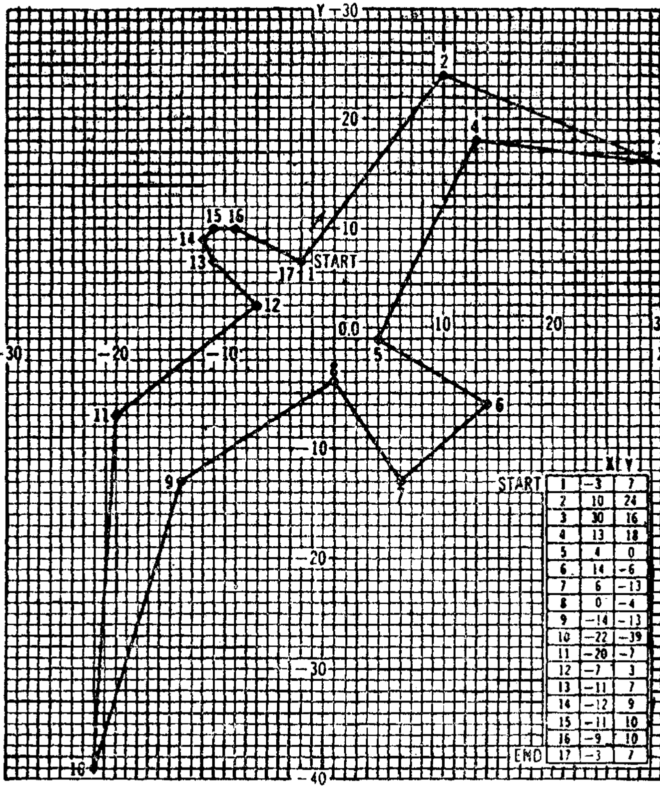
此外，因为屏上X轴和Y轴的单位长度不相等（X轴的点分布比较密，Y轴的点分布较疏），其比例关系约为1:1.212。因此，Y值还应除以比例常数1.212。

有了一幅定画造型结构图，加上一些特殊语句，便可以得到一幅动画了。

以下详细介绍这个设计过程及程序，供读者上机运行。

2. 编写鸟的绘图程序

```
10   HGR:HCOLOR=3:F=1.212
20   N=16:DIM  X(N), Y(N)
30   FOR  I=0  TO  N:READ  X(I):NEXT I
40   FOR  I=0  TO  N:READ  Y(I):Y(I)=
     -Y(I)/F:NEXT  I
50   HPLOT  140+X(0), 80+Y(0)
60   FOR  I=1  TO  N
70   HPLOT  TO  140+X(I), 80+Y(I)
80   NEXT  I
90   END
1000 DATA  -3, 10, 30, 13, 4, 14, 6,0, -14,
     -22, -20, -7, -11, -12, -11, -9, -8
```



		X	Y
START	1	−3	7
	2	10	24
	3	30	16
	4	13	18
	5	4	0
	6	14	−6
	7	6	−13
	8	0	−4
	9	−14	−13
	10	−22	−39
	11	−20	−7
	12	−7	3
	13	−11	7
	14	−12	9
	15	−11	10
	16	−9	10
END	17	−3	7

图 8—1　鸟的造型结构图

```
1500  DATA  7, 24, 16, 18, 0, -6, -13, -4,
      -13, -39, -7, 3, 7, 9, 10, 10, 7
```

3. 飞鸟的连续动作

要使鸟"飞起来"——在屏幕上移动，则可利用人的视觉暂留作用，将图形以显示—擦去—显示的方式来实现。以下

程序可完成这一工作。

```
10   HGR2:HCOLOR=3:F=1.212
15   GOSUB 500
20   P=15:J=80
30   FOR I=240 TO 25 STEP -P
40   HCOLOR=3:GOSUB 200
50   FOR D=1 TO 20:NEXT D
55   HCOLOR=0: GOSUB 200
60   NEXT I
65   P=P+2
70   GOTO 30
80   NEXT I
90   END
200  HPLOT I+X(0), J+Y(0):FOR K=1 TO
     N
220  HPLOT TO I+X(K), J+Y(K):NEXT K
240  RETURN
500  N=16:DIM X(N), Y(N)
600  FOR I=0 TO N:READ X(I):NEXT I
700  FOR I=0 TO N:READ Y(I):Y(I)=
     -Y(I)/F:NEXT I
800  RETURN
1000 DATA -3, 10, 30, 13, 4, 14,6,0, -14,
     -22, -20, -7, -11, -12, -11, -9, -8
1500 DATA 7, 24, 16, 18, 0, -6, -13, -4,
     -13, -39, -7, 8, 7, 9, 10, 10, 7,
```

在上面程序中，第500行的子程序将图形坐标 读 入。第

200行的子程序画出鸟的图形。第55行将颜色定为黑色，将刚画出的图形涂黑，程序回到第30行，重复上述步骤。由于起点不同（不同的I值），因此，会得到处于不同位置的图形。第50行造成视觉暂留的一段延时（读者可以改变D值而得到不同的显示情况）。变量J代表图形中心点的高度，变量P代表图形移动速度。第65行将图形移动的速度加快，因此，会看到鸟越飞越快。

读者会注意到，虽然利用上述的方法，可以很容易地绘出一个图形来，但对于动画来说，仍存在不少问题，例如：

（1）速度太慢——只利用FP BASIC语言进行运算，产生位置、颜色等信息，其速度太慢，因此画面变化缓慢。

（2）没有部分涂黑（消失）的功能——当两个图形部分重叠时，将一个图形擦掉，同时也会使另一个图形部分变黑。

（3）难以将画面缩小、放大、旋转等等——在FP BASIC 中复杂的坐标转换，屏幕很难适应这些要求。APPLE Ⅱ的设计者注意到上述问题，并结合绘图的常用性，提供了一个图形表(SHAPE TABLE)及几条相关语句，详见第六节介绍。

第六节 绘图技巧——图形表

首先介绍图形表的建立。当然，这是一项烦琐费时的工作，但加上一些专门语句后，会使这些努力有着特别的代价。

1．建立图形表的一般步骤如下：

1）建立图形；

2）决定图形向量；

3）将向量以十六进数表出；

4）加上标记；

5）将向量数据存入内存；

6）将向量数据由内存中取出，存入磁盘或磁带；

7）使用时，将向量数据从磁盘或磁带中取出，以供执行。

2. 图形表的相关语句

图形表的相关语句有：DRAW ；X DRAW ；ROT；SCALE。其功能、注释及用法详见附录一。

向量定义如图8-2所示。

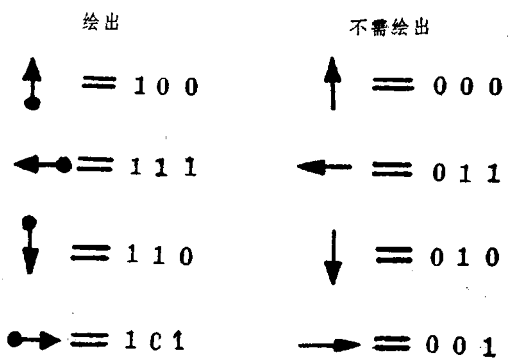


图 8-2

3. 图形表的建立过程

下面以画一辆战车为例，介绍图形表的建立过程：

（1）建立图形。如图8-8所示，用33个向量（段）构成“战车”图形。

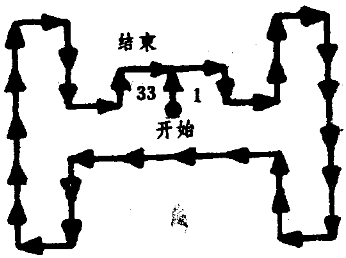


图 8-3

（2）画出向量图。如图8－4所示。

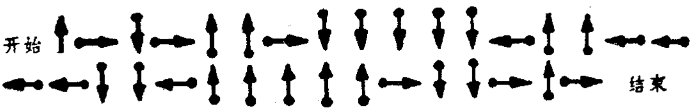
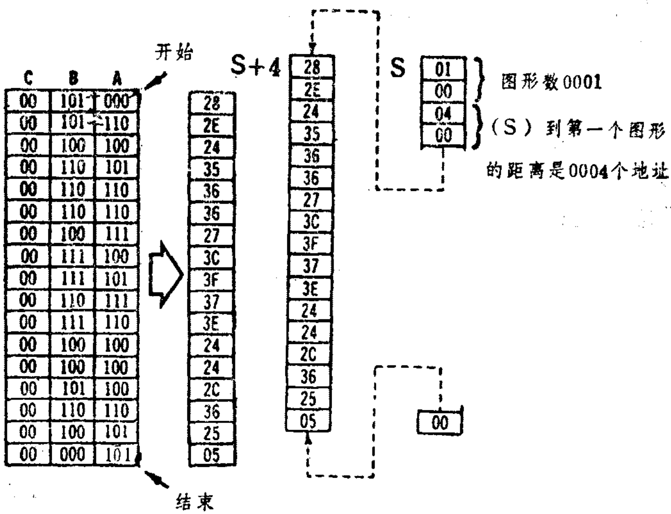


图 8－4

（3）根据向量图排出向量表，写出向量字，排好序。如图8－5的a、b、c所示。每一个字节可分成a、b、c三段来存放图形向量。



a.向量表　　b.向量字　　c.排好序

图 8-5

把向量字按一定次序输入计算机后，利用DRAW语句等即可画图。下面是一个用图形表作图的实例。

【例13】用图形表及DRAW，X DRAW，ROT，SCALE等语句，画出“方形”及“十字框”的旋转、放大/缩小、可变颜色的图形。

首先选定其图形如图8-6所示。

然后，写出向量表、向量字及编排好顺序。如图8-7所示。

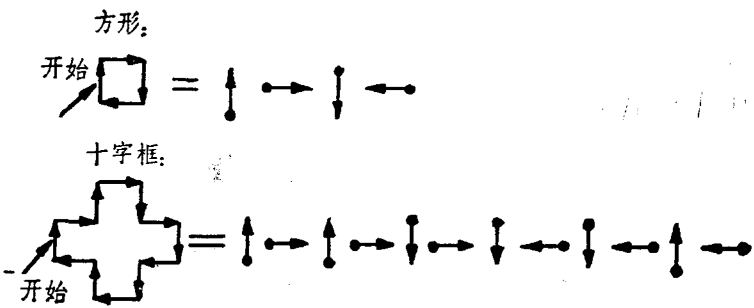
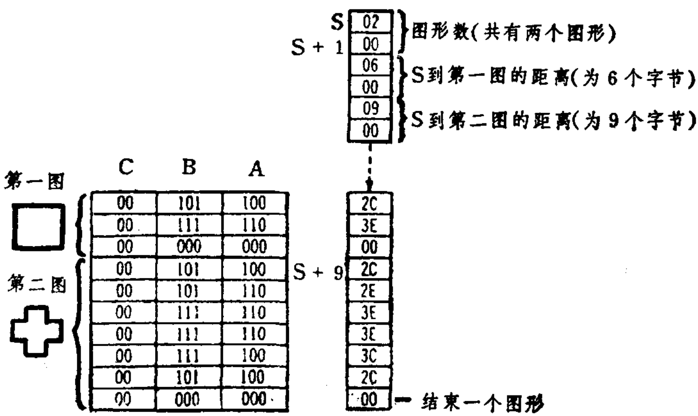


图 8-6



（a）向量表　（b）排好顺序的向量字

图 8-7

再把图形表的向量字输入内存。这可以通过监控命令的键盘输入十六进数，也可以在BASIC程序语言中用DATA语句输入十进数。

本例中两个图形的向量字，其十六进数的与十进数的关系如下：

十六进数	十进数
02	02
00	00
06	06
00	00
09	09
00	00
2C	44
3E	62
00	00
2C	44
2E	46
3E	62
3E	62
3C	60
2C	44
00	00

下面的程序可输入上述图形表，并可以用直观的屏幕选单方式，选择画出方形或画出十字框。程序中还运用了“定比例”SCALE及“旋转”ROT语句。

程序如下：

```
10  TEXT : CLEAR : POKE 216, 0 : A = FRE ( 0 )
30  GOTO  1000
50  REM  RANDOM  COLOR
52  C = INT ( RND (1) * 8 )
```

```
53 IF C=OC THEN 52
54 HCOLOR=C:OC=C
55 RETURN
60 REM * TEST KEYBOARD
70 SW=0
74 IF PEEK(-16384)>127 THEN SW=1
76 POKE -16368,0
78 RETURN
99 REM *FLASHING SCREEN
100 GOSUB 50:HPLOT 0,0:CALL 62454
105 GOSUB 70
110 IF SW=1 THEN RETURN
115 GOTO 100
199 REM *DRAW SHAPE
200 ST=-ST:GOSUB 50
205 B=1:E=R
210 IF ST<1 THEN B=R:E=1
230 FOR I=B TO E STEP ST
235 ROT=I:SCALE=I
240 DRAW SH AT X,Y
242 HCOLOR=7-C
243 X DRAW SH AT X,Y
245 NEXT I
250 GOSUB 70
255 IF SW=1 THEN RETURN
260 CALL 62454:GOTO 200
999 REM * LOAD SHAPE TABLE START
```

ADDRESS

```
1000 POKE 232,00 : POKE 233,03
1010 REM * LOAD SHAPE TABLE DATA
1020 FOR LOC=768 TO 783
1030 READ BYTE : POKE LOC,BYTE
1040 NEXT LOC
1050 DATA 02,00,06,00,09,00,44,62,00,44,46,
     62,62,60,44,00
1060 ROT=0 : SCALE=8 : X=140 : ST=1
1999 REM * MAIN DRIVER
2000 TEXT : HOME : PRINT TAB(10)"* *
     SHAPES TABLE GRAPHIC DEMO * *"
2001 PRINT TAB ( 18 ) "25/03/1984"
2002 PRINT "MENU : " : PRINT "-----"
2004 PRINT TAB ( 5 ) "1--SQUARE" : PRINT
2006 PRINT TAB ( 5 ) "2--CROSS" : PRINT
2008 PRINT TAB ( 5 ) "3--FLASHING"
     : PRINT
2010 PRINT TAB ( 5 ) "0--EXIT PROGRAM"
2020 VTAB ( 14 ) : CALL -958 : INPUT
     "WHICH ONE?=" ; N : SH=N
2025 IF N=0 THEN VTAB 4 : CALL -958:
     VTAB 12 : PRINT "GOOD BYE FOR
     NOW...." : END
2030 IF N<1 OR N>3 THEN PRINT
     CHR$ ( 7 ) : GOTO 2020
2040 HGR : POKE -16302,0
```

```
2050  CALL  62454
2060  IF  SH=1  THEN  Y=95:R=80
2070  IF  SH=2  THEN  Y=110:R=48
2080  ON  N  GOSUB  200,200,100
2090  GOTO  2000
```

该程序把所定义的图形（在第1050行定义的）显示在高分辨率屏幕上。当变量 I 的值增加时，会使图形的大小变化并旋转。

各语句的功能简介如下：

第10行	清屏幕和内存。
第50～59行	选定新的颜色值。
第69～78行	检测键盘。
第99～115行	将屏幕用上一次的颜色填满。
第199～260行	定图形的大小，绘出并且旋转它；定义负的增加值ST； 选择颜色值； 定义FOR…NEXT循环中的起始值B及终止值E； 按 I 值旋转图形； 按 I 值定图形的大小； 在X、Y处画出图形SH； 检测键盘，GOSUB 70； 用上一次的颜色将屏幕填满。
第1000～1050行	将图形表存入内存，使用的是第1050行DATA的图形表数据。
第1060行	设定ROT值，定SCALE值及X的位置。 设定ST=1。

第2000～2030行　主程序段。

第2040行　设定图形的状态。

第2050行　用上一次的颜色将屏幕填满。

第2060～2070行　设定绘图的Y位置，定出绘图范围的 R 值

第2080行　执行选择的子程序。

第2090行　返回主程序（MENU：）。

读者可输入并运行此程序。试选择“1”，看看方形的旋转和大小变化。试选择“2”，看看十字框。若选“3”，可看到闪烁翻腾的“波浪”（在彩色屏上更是漂亮）。若要结束程序的运行，可选择（键入）“0”。

第九章　DOS3.3（磁盘操作系统）使用指导

DOS是英文Disk Operating System(磁盘操作系统)的缩写。操作系统是一套为了提高计算机的工作能力，扩展计算机的功能而配备的软件。

磁盘是计算机常用的外存贮器，可分为硬磁盘和软磁盘两种，简称硬盘和软盘。一般的APPLE Ⅱ计算机，只能使用软盘作外存贮器。软盘要放入磁盘驱动器中使用，磁盘驱动器也称为磁盘机。软盘在磁盘机中是高速旋转的，它通过移动的磁头存取信息。APPLE Ⅱ微型机的一个软盘，在DOS3.3系统下有140K字节的随机存取容量。

要在APPLE Ⅱ计算机上使用磁盘，必须具备以下器件：

（1）磁盘驱动器接口板(一块板可接两台磁盘驱动器)；

（2）磁盘机（一般为两台）；

（3）磁盘操作系统，例如DOS，Pascal或CP/M及其它软件。

磁盘驱动器接口板，可插在主机上除0号外的其余七个扩展接口插座中。通常将磁盘驱动器接口板插在#6扩展插座中。关于磁盘机，磁盘驱动器接口板与主机之间的连接，在第一章中已有介绍，这里就不再重复了。

第一节　基 本 操 作

一、磁盘操作系统（DOS）的调入

凡是要利用磁盘这种外存贮器来存贮程序及数据，就必须使用DOS。DOS的作用是管理磁盘上信息的存取，控制主机和磁盘之间的信息传送。

磁盘操作系统存放在专门的一个磁盘中。在使用时，必须先将它调入内存中。从磁盘中将操作系统调入内存的过程一般称为“引导”或“导入”。

DOS调入内存后，可以通过一系列的命令进行各种操作，如存取程序，建立文件，磁盘初始化等。DOS的组织成分如图 9－1 所示。

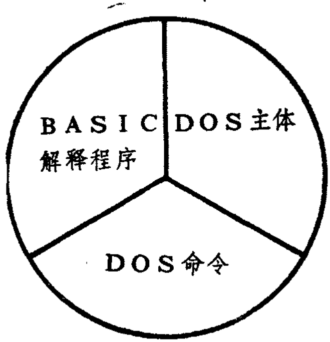


图 9-1　DOS的组织成分

使用者可以通过 DOS中的命令来使用 DOS 主体，同时也可以使用BASIC语言。

APPLE Ⅱ机对DOS的处理，要比对BASIC解释程序的处理优先。从键盘上输入的命令首先经过DOS，如果是DOS命令，DOS就调用主体中的程序进行处理。如果不是，再交给BASIC解释程序处理。处理过程如图9-2所示。

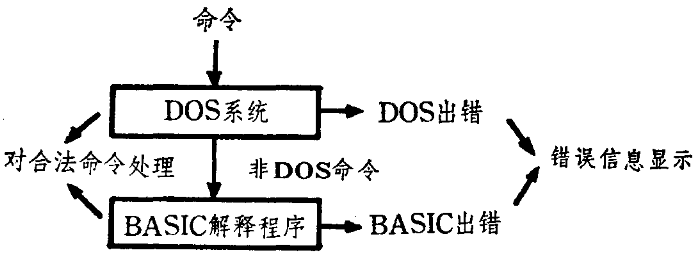


图9-2

在连接好磁盘驱动器接口板及磁盘驱动器之后，就可以从存放DOS的磁盘中把DOS调入内存了。

DOS只能从第一号驱动器中调入，因此DOS盘必须插入指定接口上的第一号驱动器中。可以用以下的方法从磁盘中调入DOS。

(1)开机直接调入。将DOS磁盘插入第一号驱动器中，然后开机，即可自动调入DOS系统。

(2)在整数BASIC或浮点BASIC状态中，可以分别用“IN#n”或“PR#n”命令调入DOS系统，其中n表示磁盘驱动器接口板所在的扩展接口的编号。

如：]PR#6　　或　>IN#6

(3)在监控状态中，可以用

*Cn00G

*nCTRL-P

这两种方式调入DOS，其中n的含义同上。如n=6，则为

*C600G （转到C600执行接口板上ROM的程序）

*6CTRL-P （在6和RETURN 中间加入了“CTRL-P”组合键）

如果第一次系统引导不成功，不妨再试一次，要注意将磁盘放好。在开机直接调入的情况下，如果主机上接的磁盘机接口板不止一块，则将从接口编号较大的接口板上的第一号磁盘机中调入DOS。

DOS调入内存后约占用10.5K字节。DOS存于内存RAM的顶部，48KRAM的APPLE Ⅱ机调入DOS后，内存分配如图9-3所示。

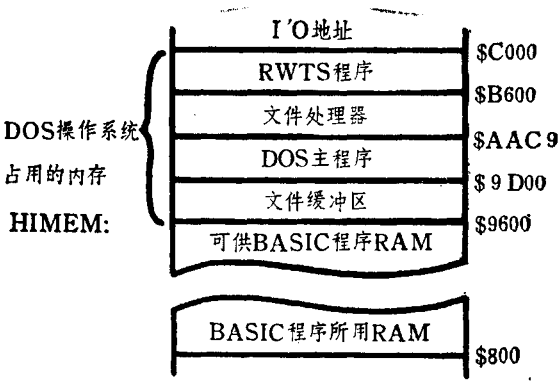


图 9-3 调入DOS后，内存的分配情况

二、磁盘的复制

使用DOS系统盘中的复制程序，可以将DOS格式磁盘上的内容，全部抄到另一个磁盘上去。比如，可以将DOS系统

盘复制一个副本，平时使用这个副本而将源盘保留下来。对于其它的磁盘也可以这样做，以免因发生意外而将宝贵的资料破坏贻尽。

在DOS3.3系统磁盘（System Master）中，用于复制磁盘的程序有两条，即COPY和COPYA，两者均可在BASIC状态下使用。另外还有一条叫COPY·OBJ0的程序，是支持上面两条程序的机器语言程序。

在使用COPY程序之前，应先明确以下概念：被复制的磁盘称为“源盘”（ORIGINAL），而复制的副本称为“复制盘”（DUPLICATE）。

复制盘无须经过“初始化”，因为COPY（或COPY A）程序会自动完成这项工作。

在进行复制之前，请将源盘的“写保护”缺口封好，以免不慎把它上面的程序擦去，而使复制工作无法进行。

目前在用的（隐含的）接口板及驱动器，称为“自然”的或“省缺”的（DEFAULT）接口板及驱动器。

现在请将系统主磁盘放入当前选择的驱动器，如S6，D1中。

打入命令：

]RUN COPYA

清屏后会显示：

APPLE DISKETTE DUPLICATION PROGRAM

ORINGAL SLOT：DEFAULT = 6

表示当前源盘用的 #6 接口上的接口板。如果使用者认为不对，只要按下相应的数字键，就可以改变。

接着按下RETURN键，则显示：

DRIVE:DEFAULT = 1

表示源盘在第一号驱动器中。

再按RETURN键，则显示：

DUPLICATE SLOT:DEFAULT=6 表示复制磁盘用#6接口上的接口板。

再按RETURN键，则显示：

DRIVE:DEFAULT=2 表示复制磁盘用第2号驱动器。

再按RETURN键，则显示：

--PRESS 'RETURN' KEY TO BEGIN COPY--

这是提示使用者，如果再按下RETURN键就开始复制了。

此时可将DOS系统盘从第1号驱动器中取出，放入需要复制的源盘，运行COPY程序就会开始复制磁盘。复制过程是首先从源盘中读取信息，然后格式化复制副本，并将信息抄入复制盘中，经过五次读、写步骤就复制好了。这时将会见到：

DO YOU WISH TO MAKE ANOTHER COPY?

（需要再次复制吗？）

如果打入Y（Yes，需要），则会出现：

--PRESS ANY KEY TO BEGIN COPY--否则，打入N（NO，不需再复制），那么将结束程序，退到BASIC状态。

按照上面的步骤，可以将DOS系统盘复制一个副本，其它使用DOS操作系统的磁盘也可以用COPYA或COPY程序复制。CP/M系统和Pascal系统也都提供了本身的磁盘复制程序。但是，一些常用的软件磁盘却无法用COPY或COPYA程序复制，原因是软件公司为了本身的利益，将磁盘上信息

的存贮方式、标记、同步方式作了修改，使正常的DOS无法识别。对于这些软件，如APPLE FORTRAN FORT2、APPLE CILLIN（诊断程序）、APPLE WRITER Ⅱ（文字处理）、DB MASTER（数据处理）及 VISICALC、VISIFILE 等和许多游戏，则只能采用更高级的复制程序进行复制。这些程序数量不少，功能较强的有COPY Ⅱ PLUS 4·×（×代表一数字、为版本顺序号）、LOCKSMITH 4·×和5·0、NIBBLES AWAY Ⅱ、CRAZY COPY、BACK-IT-UP Ⅱ等，这里就不一一介绍了。此外，还可以用硬件复制方法，将内存中的程序抄到磁盘中。这些硬件插板有 WILDCARD、WILDCARD PLUS、ALASKA CARD及REPLAY Ⅱ等等。

第二节　DOS系统的基本命令

这一节将介绍DOS系统的基本命令及其操作过程。

一、DOS命令的基本格式

```
 INIT      HELLO, S6,  D2,    V128
 ︶︶        ↓        |    |       |
命令字     文件名      |    |       └→磁盘的册编号
                      |    └→磁盘机号
                      └→磁盘驱动器接口板所在的接口
                        编号
```

这是一个典型的DOS命令，它包括下列三种内容：

（1）命令字。这是DOS定义的，每个DOS命令都只有一个命令字组合。

（2）文件名。这是磁盘中文件的名称，文件就是作为一个整体使用的相关数据的集合，如一条程序，一批数据等

都是一个文件，不同的文件按照各自的名进行区分。

（3）接口板编号、磁盘机号、磁盘的册编号。这一部分用于说明命令对象（文件或磁盘）的位置（在哪一个磁盘上）。它们常常可以省略，或部分省略，如果省略了，则被认为是与上次使用过的相同。通常称这种规律为“省缺”。

一般来说，DOS命令可以按以下方式表示：

INIT f〔，Ss〕〔，Dd〕〔，Vv〕

其中方括号的内容是可以省略的，字母的含义为：

f——文件名，由1～30个字符组成。其中第一个字符必须是字母，但也可以是∧、@、〕、〔等几个符号。文件名中不能有逗号；

s——磁盘机接口板所在的接口号码，为1～7的任一数字；

d——磁盘机号码，为数字1或2；

v——磁盘册编号，是1～254的任一整数，0和省略等效，但在INIT命令中被认为是254。

二、基本DOS命令

1. INIT f〔，Ss〕〔，Dd〕〔，Vv〕

这个命令的功能是对一个磁盘进行初始化，在磁盘上划分出磁道和扇区，并写入DOS操作系统。每个新的磁盘在使用之前都必须先进行初始化，经DOS3.3初始化后的磁盘共有35条磁道（$0—$22），每条磁道分为16个扇区（$0—$F），每个扇区可以存贮256字节的信息。整个磁盘的容量为140K字节，其中DOS占用第0，1，2三条磁道，第$11（十进制17）条磁道用于存放文件目录，留给用户使用的空间为124K字节。

2. CATALOG 〔,Ss〕 〔,Dd〕

打入这个命令后，DOS会将磁盘中文件的目录显示在屏幕上。这是一个经常使用的命令，文件名的显示格式为：

DISK　VOLOME　254　第一行显示磁盘册编号。

```
* A   006   HELLO→文件名
| |      |→文件件的大小（以扇区为单位）
| |→文件的类型，A为浮点BASIC文件；I为整数
|      BASIC文件；B为机器语言文件；T 为文本文件
|→上“锁”标记
```

3. SAVE　f　〔,Ss〕　〔,Dd〕　〔,Vv〕

功能：将内存中文件名为f的BASIC程序存入磁盘。

4. LOAD　f　〔,Ss〕　〔,Dd〕　〔,Vv〕

功能：将磁盘中文件名为f的BASIC语言程序调入内存。

5. RUN　f　〔,Ss〕　〔,Dd〕　〔,Vv〕

功能：把磁盘中文件名为f的BASIC 程序文件调入 内存并马上执行。

6. DELETE　f　〔,Ss〕　〔,Dd〕　〔,Vv〕

功能：删除磁盘中无用的文件，其中 f 为要 删 除 的 文件名。

7. RENAME　f_1, f_2　〔,Ss〕　〔,Dd〕　〔,Vv〕

功能：给磁盘文件换名，其中f_1为旧文件名，f_2为新文件名。

8. VERIFY　f　〔,Ss〕　〔,Dd〕　〔,Vv〕

功能：检验磁盘中文件名为f的文件是否可读。

9. LOCK　f　〔,Ss〕　〔,Dd〕　〔,Vv〕

功能：锁住文件名为f的文件。文件被锁住后，就 不 能被删除（DELETE）或换名（RENAME）了。

10. UNLOCK f 〔,Ss〕 〔,Dd〕 〔,Vv〕

功能：把文件名为f的文件解锁。

以上简要地介绍了基本的DOS命令，读者如需要更详细的了解，可以参阅APPLE Ⅱ微型计算机的《磁盘操作系统手册》。

三、基本DOS命令的操作过程

1. 磁盘初始化

新的磁盘必须经过DOS系统初始化后，才能用来存贮程序或数据。旧的磁盘（已经过初始化，并录有程序，但不一定是DOS系统的）如果重新初始化，将会失去原来的全部信息。一个磁盘经过DOS系统初始化后，上面就已经录有DOS系统及一条BASIC（一般为 FP BASIC）的“欢迎”程序，这个磁盘上还有496个扇区（包括“欢迎”程序占用的扇区）即124K空间供用户使用。

磁盘初始化的操作过程如下：

准备好DOS系统磁盘及空白磁盘各一只。

(1) 将DOS系统从系统盘调入主机。

(2) 打入NEW命令，清除原来的“HELLO”程序，然后输入新的“HELLO”程序，如：

```
10 REM GREETING PROGRAM
20 TEXT: HOME
30 PRINT "DOS 3.3 DISK CREATED ON
   APPLE Ⅱ"
40 PRINT "BY X·Y·WANG, 10—MAY—1983
50 PRINT "FOR DOS BASIC PROGRAM
   STORAGE"
60 END
```

读者也可以输入别的程序，试运行一次认为满意后，继续以下操作。

（3）将空白磁盘放入第2号驱动器，打入：

INIT　HELLO，D2

如果只有一台驱动器，则可将系统磁盘取出，换为空白磁盘，然后打入INIT　HELLO，系统就会对该磁盘进行初始化，大约30秒后，光标重新出现，表示初始化过程完毕。打入CATALOG命令，可以看到：

DISK　VOLUME　254

A　002　HELLO

]

如果读者要自己定义册编号，可把上面的INIT命令改为：

INIT　HELLO，D2，V×××　其中×××为1～254之间的整数。当然，“欢迎”程序的名字也可以不用HELLO。

（4）试从刚初始化的磁盘中调入DOS操作系统。

2．DOS其它命令的操作

将已经初始化的磁盘放入第一号驱动器中，打入命令：

CATALOG

这样，可以见到磁盘的册编号及磁盘中文件的目录（包括文件名、文件的类型及文件的长度）。这个命令是DOS最常用的命令。如果接着打入命令：

LOAD　HELLO

LOAD命令会将磁盘中的“HELLO”程序载入内存（可以用LIST命令检验）。如果再打入

SAVE　TEST

这个命令将内存中文件名为TEST的BASIC程序存入磁

盘。这时如果再用CATALOG命令检查磁盘，可见到盘中已录有两个文件：HELLO 及 TEST。由于当前机器内存中只存有如上所述的“欢迎”程序，因此这两个文件的内容是一样的，只是文件名不同。但DOS系统认为，这是两个彼此无关的文件。这时如要将文件TEST的文件名换为 TESTI，可以打入

RENAME TEST，TESTI

再用CATALOG命令检查，可以发现原来的TEST 已变为TESTI。现在，可用LOCK命令将这两个文件锁起来。

LOCK HELLO

LOCK TESTI

这时如果再用CATALOG命令检查，会发现文件名前已有“*”号，表示该文已上锁。此时，如打入命令：

DELETE TESTI

机器会给出“FILE LOCKED”信息，表示文件 TESTI被锁住，不能删去。

如果要将TESTI解锁，可以打入命令：

UNLOCK TESTI

此时如用CATALOG检查，TESTI前的星号已被清去。如果继续打入命令：

DELETE TESTI

此时再来查看磁盘的目录，就会知道TESTI 已经被删去了。

当一个文件被删去后，DOS会将这个文件所占用的磁盘空间重新“释放”出来。

在使用 SAVE 及 RENAME 这两个命令时应该注意，DOS在这两个命令的处理上是有缺点的。

在使用SAVE命令时，DOS首先查看磁盘上的文件目

录，要是命令中的文件名已经存在，而且文件类型也相同，则旧的程序将会被新的程序所取代，也就是“覆盖”了旧的程序。在这种情况下，如果新的程序比旧的长，则不会有什么问题；要是旧的程序比新的长，就有问题了，因为DOS不会将旧文件中多占的空间（扇区）释放出来。当存入较短的同名新文件后，再查看磁盘的文件目录时，就会发现新文件仍占有和旧文件相同的扇区。这样不仅浪费了磁盘空间，而且还将得到错误的长度信息。因此，在这种情况下，应当先用DELETE命令将旧的文件删除掉，释放出全部扇区，然后再存入新的文件。当SAVE命令中的文件名在磁盘中存在而文件类型不同时，会产生出错信息：FILE TYPE MISMATCH（类型不符）。

在用RENAME 命令为文件换名时，DOS不管新文件名在磁盘中是否已经存在，这样就有可能产生重名文件。解决的办法是：只能再用RENAME 命令，将重名的文件名换为别的文件名。

第三节 机器语言的DOS命令

本节介绍在DOS系统中用磁盘存取机器语言文件的方法。

一、机器语言的DOS命令

在DOS系统中有三个用于处理机器语言的命令。

1. BSAVE f, Aa, Ll〔, Ss〕〔, Dd〕〔, Vv〕

功能：将指定区域内的段机器码（二进制数）存入磁盘。A是区域的起始地址，L是区域的长度（以字节为单位）。a在0～65536之间取值，l在0～32767之间取值；a

和1均可以采用十进制数或十六进制数表示，在用十六进制数时要加上“$”号。如果一个机器语言的程序大于32767字节，就要用两次BSAVE命令分别贮存。

BSAVE命令也存在和SAVE命令相同的缺点，这可以用与SAVE命令相同的办法解决（参见上一节SAVE命令）。

2. BLOAD f〔, Aa〕〔, Ss〕〔, Dd〕〔, Vv〕

功能：从磁盘中取出二进制文件放入内存。其中位置参数A可以省略，如省略，则表示该文件的取出后存放地址与写入时的地址相同。

如果机器语言程序没有存入适当的位置，则该程序很可能不会正常工作，除非程序中没有分支和转移（如JMP、JSR）。

3. BRUN f〔, Aa〕〔, Ss〕〔, Dd〕〔, Vv〕

功能：将程序BLOAD入内存后，再跳到起始地址去执行一个JMP a的命令。入口地址和起始地址不同的机器语言程序不能使用这个命令。

二、机器语言DOS命令的操作过程

1. 将机器语言的程序或数据存入磁盘。如：

BSAVE BINARY FILE, A$3D0, L$30

上面的命令表示：将DOS的第三页向量表存入磁盘，起始地址是$03D0，程序长度为$30字节。这个命令中的起始地址及长度也可用十进制数表示：

BSAVE BINARY FILE, A976, L48

用下面的命令可以把高分辨率图形存入磁盘，如：

BSAVE HGR·PIC A$2000, L$2000

该命令可以把第一页高分辨图形存入磁盘。以下的命令和这个命令是等价的：

BSAVE　HGR•PIC, A $ 2000, L8192

其中长度采用十进制数表示，起始地址仍用十六进制数表示。

2. 将机器语言程序或数据从磁盘取出，存入内存。如：

BLOAD　HGR•PIC

这个命令可将刚存入磁盘的高分辨率图形取出，仍然放在第一页高分辨率图形区域（图不指定起始地址，所以从写入磁盘时的起始地址开始存放）。

BLOAD　HGR•PIC, A $ 4000

这个命令可以将 HGR•PIC 图形放入内存第二页高分辨率图形区中去。

BLOAD 命令只清除机器码占用的内存地址单元，而不影响其它内存单元的内容。因此，可以将机器语言程序作为BASIC程序的子程序使用。这对于某些方面，如过程控制等，是十分方便的。

3. 将机器语言程序载入内存并执行。如：

BRUN　FID

执行系统磁盘中的FID程序，该程序的起始地址是 $ 803（2051），这个命令相当于：

BLOAD　FID

CALL　2051　（或 * JMP　$ 803）

因此，BRUN命令只适用于那些入口和起始地址相同的机器语言程序，因为它是从起始地址开始执行的。又如：

BRUN　FID, A $ 6000

这个命令将FID程序从磁盘中取出，从内存地址 $ 6000 开始存放（而不是 $ 803），然后再JMP　$ 6000。

第四节　DOS命令在程序中的使用

一、基本 DOS 命令在程序中的使用

请看这个例子：

```
10  REM  USING  DOS  COMMAND
20  D$ =  CHR$(4): REM  CTRL-D
30  PRINT  D$; "CATALOG"
40  PRINT  D$; "RUN  PROGRAM1"
50  END
```

在这程序中，第30句及第40句使用了DOS 命令：CATALOG 及 RUN。使用的方法是：PRINT语句打出一个以字符"CTRL-D"为首的字符串（在浮点 BASIC 状态下，可用CHR$(4)代替），字符串后为DOS命令。"CTRL-D"可以通过按着 "CTRL" 键再按下 "D" 键得到。由于整数BASIC无CHR$函数，因此只能用上述第一种方式获得"CTRL-D。" "CTRL-D" 和其它"CTRL-"字符一样，是不显示的。

下列语句都使用了DOS命令，尽管方法不同，但作用是一样的（其中D$ =CHR$（4）或D$ =" "，后者两个引号之间插入了"CTRL-D"）。

```
100  PRINT  CHR$ ( 4 ); "CATALOG"
100  PRINT  D$"CATALOG"; ", S"; S; ",
     D"; D
100  PRINT  "  "; "CATALOG"
100  PRINT  "CATALOG, S6, D1"
```

在最后一句中，引号和字母C之间插入了"CTRL-D"。

例如有一程序段：

```
50  PRINT "THIS DISK CONTAINS FOLLO-
    WING FILES: ";
60  PRINT CHR$(4); "CATALOG"
```

这个程序段并不能列出磁盘上的文件目录，因为第50句后面的分号使第60句中的"CTRL-D"不能成为一行中的第一个字符，可以将第50句后的分号去掉，或将第60句改为：

```
60  PRINT CHR$(13)+CHR$(4); "CATALOG"
```

CHR$（13）即RETURN，这样便使得"CTRL-D"（CHR$（4））成为第一个字符了。

通常，我们可以令：D$ = CHR$ (13) + CHR$ (4)。这一技巧可以用于跟踪命令调试程序，因为TRACE命令打印出的行号会使"CTRL-D"不是一行中的第一个字符。

二、其它DOS命令

1．MON和NOMON

MON命令使DOS进入监视状态，可监视DOS命令的执行及磁盘上信息的输入和输出。NOMON命令则使DOS取消监视状态。

MON和NOMON命令的格式为：

MON 〔C〕 〔, I〕 〔, O〕

NOMON 〔C〕 〔, I〕 〔, O〕

其中三个参数的含义为：

C——监视DOS命令（PRINT "CTRL-D"形式的DOS命令）；

I——对从磁盘上输入（用READ命令输入）的信息进行监视；

O——对输出到磁盘的信息（用WRITE命令输出的数

据）进行监视。

MON和NOMON命令至少要与上述三个参数中的一个参数连用，只打入MON或NOMON，DOS是不予理会的。

MON的作用一直持续到执行：INT、FP、RESET、CALL 976、NOMON等命令为止。

在MON命令中，参数的排列次序与功能无关，且参数之间的逗号可以省略。

初学者在学习使用文本文件时，最好都加上MON命令。

2．MAXFILES命令

MAXFILES命令的功能是确定可以同时打开的文本文件的最大数目。MAXFILES命令的格式为：

MAXFILES　n

其中n在1～16中取值。MAXFILES命令为DOS开辟n个文件缓冲区，每个缓冲区为595个字节。

当DOS被调入内存时，自动执行MAXFILES 3，即保留三个文件缓冲区（参见图9－3）。

当MAXFILES的数目改变时，HIMEM：指针也同时改变，因此整数BASIC程序或浮点BASIC程序的字符串变量将被清除。所以在浮点BASIC程序中改变MAXFILES的值时，MAXFILES命令必须是程序中的第一句；而在整数BASIC程序中则不能改变MAXFILES的数目。

第五节　文本文件的使用

DOS系统可支持两种供BASIC语言使用的文件——顺序存取文件及随机存取文件（简称顺序文件和随机文件）。

一、顺序存取文本文件

顺序文件只能按顺序从文件的头一个记录起，逐个记录地进行存取，而随机文件可以直接存取文件中任一记录。顺序文件与随机文件两者各有利弊。至于应在程序中使用哪一种文件，则要依照具体情况而定。

顺序文件及随机文件的逻辑结构如图9-4所示。

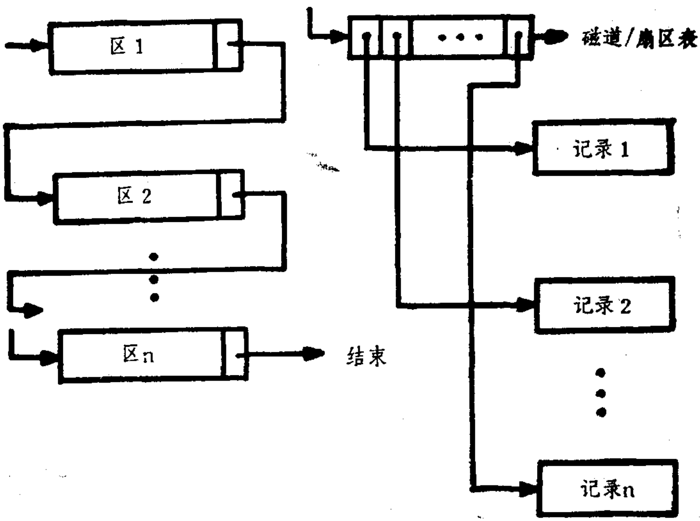
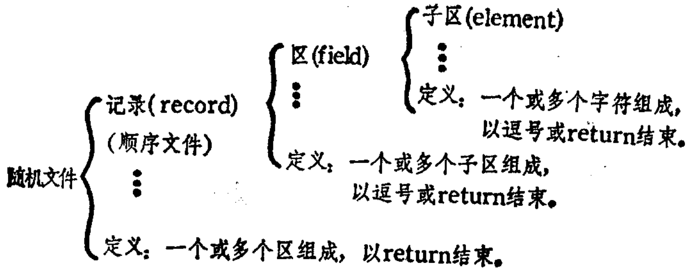
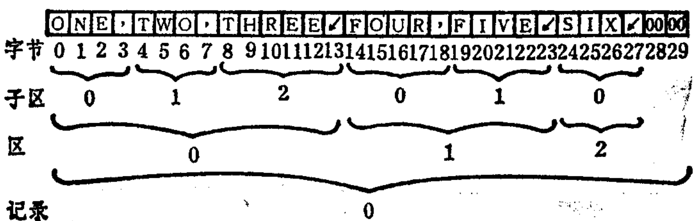


图 9-4　顺序文件及随机文件的逻辑结构

DOS 3.3的文件结构如下，其中顺序文件可以看成是随机文件的一个记录。



【例 1 】一个记录的结构，其记录长度L＝80字节。



使用顺序文本文件的命令有下列六种。

1. OPEN f〔, Ss〕、〔, Dd〕 〔, Vv〕

功能：打开文件。在对一个文本文件进行读写操作之前，必须先将该文件“开启”或称为“打开”。OPEN命令为文件分配一个缓冲区。如果名称为f的文件已开启，则先关闭该文件，然后再打开；如果磁盘中没有该文件，则新建立一个。OPEN命令必须在程序中使用。

【例 2 】100 PRINT D$; "OPEN TEXT FILE"

2. CLOSE 〔f〕

功能：关闭文件，并释放该文件占用的缓冲区。CLOSE

命令有两种形式：如果命令中没有文件名，则关闭所有打开的文件；若有文件名，则只关闭指定的文件。

3. WRITE　f

功能：执行写操作。该命令将后面PRINT命令的内容写入文件f中，而不是送屏幕显示。出错、程序中的DOS命令或PRINT"CTRL-D"、INPUT等命令，都会取消WRITE命令。

WRITE命令只能在程序中使用。

【例3】300　PRINT　D$；"WRITE　TEST DATA"

4. READ　f

功能：执行读操作。该命令将文件中的内容读入BASIC变量中。READ命令执行后，以后的INPUT命令将从磁盘文件中（而不是从键盘上）接收数据。DOS命令或PRINT "CTRL-D"命令会取消READ命令。READ命令只能用在程序中使用。

【例4】60　PRINT　D$；"READ　DATA1"

5. APPEND　f　〔，Ss〕　〔，Dd〕　〔，Vv〕

功能：将数据加在已存在的顺序文件后面，作为该文件的延续。APPEND命令只能和WRITE命令连用。使用APPEND命令时，不必将文件开启（OPEN），因为APPEND命令会自己完成开启文件的操作。APPEND命令甚至可在已经锁住的文本文件后增加数据。APPEND命令只能在程序中使用。

6. POSITION　f〔，Rp〕

功能：指定文件读写指针的位置。因此可用该命令读出指定区中的数据。其中Rp代表指针从当前位置向前移p个区

（两个回车命令之间的内容称为一个区）。p的值必须是0～32767之间的整数；如果p＝0，则读写指针位置不改变。

POSITION命令象其它DOS命令一样，会把READ或WRITE命令取消。因此，它必须用在READ或WRITE命令之前，或者在使用POSITION命令后再补回READ或WRITE命令。

二、顺序文本文件的使用方法

1．顺序文件的建立

在DOS系统中，建立一个顺序文件的基本方法是：

（1）用OPEN命令打开一个文件；

（2）用WRITE命令对该文件进行“写”操作；

（3）使用PRINT命令向文件输出资料（数据）；

（4）最后用CLOSE命令关闭该文件。

【例5】建立一个名为“PHYSICS”的顺序文本文件。

```
100  REM  MAKE  PHYSICS
110  D$ =CHR$(4): REM  CTRL-D
120  PRINT  D$; "OPEN  PHYSICS"
130  PRINT  "ENTER  DATA  IN  THE
     FORMAT: "
140  PRINT  "STUDENT'S  NAME, STUDENT
     NO., SCORE"
150  PRINT  "WHEN  FINISH  TYPE  0, 0,
     0"
160  I=I+1
170  PRINT  "#"; I; ": ";
180  INPUT  " ", N$, N, G
190  IF  N>50  OR  N<0  THEN  260
```

```
200 IF G<0 OR G>120 THEN 260
210 IF N$="0" THEN GOTO 280
220 PRINT D$;"WRITE PHYSICS"
230 PRINT N$; ","; N; ","; G
240 PRINT D$
250 GOTO 140
260 PRINT "INVALID INPUT, RETYPE
    PLEASE"
270 GOTO 140
280 PRINT D$;"CLOSE PHYSICS"
290 END
```

该程序将建立一个名为“PHYSICS”的顺序文本文件，该文件包括了学生姓名、学号、考试成绩的资料，科目是“PHYSICS”（物理），其中学生人数少于50人，满分为120分。

读者可以利用本章末实习思考题9的数据作为输入资料，建立一个“PHYSICS”文本文件。

例5中部分语句含义为：

第120句：打开文件；

第180句：从键盘上取得数据；

第220句：进行写操作；

第230句：将数据写入文件中，注意格式；

第280句：关闭该文件，结束资料输入。其中第240句用于取消220句的“写”命令，防止前面的提示信息被写入文件中去。

这是一个简单的程序，它不足之处在于：

（1）不能对输入的数据（在未写入磁盘之前）进行修改；

(2)不能防止写入重复的资料。

2. 顺序文件的读出

读出一个顺序文件的基本步骤为：

(1)打开该文件(OPEN)；

(2)对该文件进行读操作(READ)；

(3)用INPUT命令从文件中读入数据；

(4)关闭该文件(CLOSE)。

如想读出前面建立的文件，可以用以下的程序：

```
100  REM  RETRIEVE  PHYSICS
110  D$ =CHR$(4)
120  PRINT  D$; "OPEN  PHYSICS"
130  PRINT  D$; "READ  PHYSICS"
140  INPUT  N$, N, G
150  PRINT  "NAME: "; N$
160  PRINT  "STUDENT  NO.: "; N
170  PRINT  "SCORE: "; G
180  FOR  I=1  TO  1000: NEXT  I
190  GOTO  140
200  PRINT  D$; "CLOSE  PHYSICS"
210  END
```

这个程序将文件"PHYSICS"中的资料读出并显示在屏幕上。程序最后在第140行以"END OF DATA"出错结束，为避免这种情况，可以加上：

```
105  ONERR  GOTO  200
```

3. 从顺序文件中寻找资料的方法

从顺序文件中寻找相应的资料，最简单的方法是：

(1)从键盘上输入要寻找的资料信息；

（2）打开文件；

（3）读顺序读文件中的每一项；

（4）和从键盘上输入的信息作比较；

（5）作出是否找到的判断；

（6）若找到，则退出(关闭文件，显示资料)；

（7）若找不到，则判断文件是否结束(出错)，若找到，则回到（3）；

（8）关闭文件，显示出"找不到"信息。

例如，要想按照一给出的学号，从"PHYSICS"文件中找出该学生的成绩，可以用以下的程序：

```
100  REM  FIND  A  RECORD
110  ONERR  GOTO  800
120  HOME
130  D$ = CHR$(4)
140  INPUT  "STUDENT  NO.:"; NI
150  IF  NI<1  OR  NI>50  THEN  PRINT
     CHR$(7); "INVALID  INPUT, REENTER
     PLEASE": GOTO  140
160  PRINT  D$; "OPEN  PHYSICS"
170  PRINT  D$; "READ  PHYSICS"
180  INPUT  N$, N, G
190  IF  NI=N  THEN  GOTO  210
200  GOTO  180
210  PRINT  "STUDENT  NO.#"; NI
220  PRINT  "SCORE: "; G
230  PRINT  "STUDENT'S  NAME:"; N$
240  PRINT  D$; "CLOSE  PHYSICS"
```

```
250 PRINT :PRINT "LOOKING FOR
    ANOTHER?";
260 GET K$:PRINT K$
270 IF K$="Y" THEN 180
280 IF K$< >"N" THEN 260
290 GOTO 999
300 PRINT CHR$ (7); "CAN'T FIND"
310 GOTO 140
999 END
```

在程序中，第140句从键盘上取得学号；第160句打开文件；第170、180句从文件中读入数据；第180句将输入学号和文件中的学号进行比较，看是否找到该学号，若找到，则显示出成绩，找不到则出错（END OF DATA）；第110句中定义出错则转向第300句；第300句显示出“找不到”信息，然后回到第140句等待输入学号。

4．在顺序文件后面增加资料的方法

在顺序文件后面增加资料的方法是：

（1）用APPEND命令指示增加资料；

（2）用WRITE命令定义“写”操作；

（3）用PRINT命令将资料加在文件后面；

（4）关闭文件，结束。

注意：APPEND命令必须和WRITE命令联用。

【例6】在上述“PHYSICS”文件的后面增加资料。

为此，只需将上述“PHYSICS”文件的第100句MAKE PHYSICS改为100 REM APPEND PHYSICS；将第120句改为：120 PRINT D$；"APPEND PHYSICS"，并增加：125 PRINT"APPENDING DATA"即可。

三、EXEC文件

EXEC文件是可以执行的顺序文本文件。EXEC文件的内容，是DOS或BASIC语言的命令或语句。当执行EXEC文件时，文件的内容会被自动地逐区读出，就象用户从键盘上输入一样。

建立EXEC文件的方法和建立普通的顺序文件方法基本相同，但在EXEC文件中用PRINT命令输出的只能是DOS命令或BASIC语句或命令。EXEC文件建立后，可用"EXEC 文件名"命令执行之。EXEC文件的使用方法，可以参考DOS系统盘的EXEC DEMO程序。

EXEC命令的格式为：

EXEC f〔, Rp〕〔, Ss〕〔, Dd〕〔, Vv〕

其中Rp表示越过P个区后开始执行。

【例7】建立一个EXEC文件。

```
100  REM  MAKE  EXEC
110  D$ =CHR$(4)
120  PRINT  D$; "OPEN  DEMO"
130  PRINT  D$; "WRITE  DEMO"
140  PRINT  "CATALOG, D1"
150  PRINT  "FOR  I=0  TO  3000:NEXT  I"
160  PRINT  "LOAD  MAKE  PHYSICS"
170  PRINT  "120  PRINT  D$; "; CHR$
       (34); "APPEND  PHYSICS"; CHR$(34)
180  PRINT  "125  PRINT"; CHR$(34);
       "APENDING  DATA",  CHR$(34)
190  PRINT  "100  REM  APPEND  PHYSICS"
200  PRINT  "SAVE  APPEND  PHYSICS"
```

```
210 PRINT "CATALOG"
220 PRINT "FOR I=0 TO 3000:NEXT I"
230 PRINT "RUN"
240 PRINT D$;"CLOSE DEMO"
999 END
```

使用EXEC文件时应注意：

（1）EXEC命令不能用CTRL-C或RESET中止；

（2）不能同时使用二个或二个以上EXEC命令。如果一个EXEC文件中含有EXEC命令，当执行这个命令时，原EXEC文件被关闭；

（3）如果一个由EXEC命令执行的程序被中断，则EXEC文件会被关闭。要是这样的程序中有INPUT语句，则该语句将读入EXEC文件的下一个区；如果这个区的内容是一个立即型的DOS命令，它就会马上被执行。其后的结果就难以预料了；

（4）如果EXEC文件执行时，文件中有一个区出错，则将继续执行下一个区。这一特点可以用来查看任意一个顺序文件的内容，先进入MON状态，然后用EXEC命令执行该文件；

（5）执行EXEC命令时，占用一个文件缓冲区。

前面介绍过将 MAKE PHYSICS 程序改为 APPEND PHYSICS程序的方法。这个例子将建立一个名为“DEMO”的EXEC文件，这个文件将会自动完成这项工作。

当输入这个文件并执行后，打入EXEC DEMO，这时机器会显示磁盘中文件的目录（即第140行语句的功能），稍候便将 MAKE PHYSICS 从磁盘上取出（假设它和DEMO存在同一张磁盘上），然后作一些修改（170行～190

行）。修改完毕后，将程序以“APPEND PHYSICS”为文件名存入磁盘，再显示磁盘文件目录，最后运行内存中的APPEND PHYSICS程序（230行）。

四、随机存取文本文件

在前面已介绍了顺序文本文件的使用方法，可能读者已经注意到顺序文本文件有两个很明显的缺陷：

（1）检查比较麻烦。如果用户要寻找其中一个记录，则必须从第一个记录开始读，直至找到要寻找的记录为止（有时POSITION命令可以帮点忙）；

（2）资料难以更新。如果用户要修改其中一个记录，则必须将整个文件从磁盘中读出，修改后再存回磁盘中。

随机存取文本文件可以避免以上的缺陷，但是它要占去较多的磁盘空间。

使用随机文件的命令和使用顺序文件的命令基本相似。

使用随机文件的命令有下列两种。

1. OPEN f, Ll [,Ss] [,Dd] [,Vv]

打开一个随机文件，其中Ll为文件记录的长度（1个字节），l为1～32767之间的整数。

2. WRITE f [,Rr]

READ f [,Rr]

对一个指定的记录r进行“写”或“读”操作，若省略Rr，则r被认为是零。

五、随机文本文件的使用方法

1. 随机文件的建立

在DOS系统中，建立一个随机文件的基本步骤如下：

（1）用OPEN 命令打开一个随机文件，命令中应定义文件记录的长度。如 OPEN 文件名，L20；（记录长度为

20个字节）

（2）用WRITE命令对一个记录进行“写”操作，命令中要包括记录号；

（3）将资料用PRINT命令送入记录中去；

（4）写完所有的记录后，关闭该文件。

【例8】建立一个库存货物的文件。

```
100  REM  MAKE  INVENTORY
110  D$ =CHR$ ( 4 )
120  PRINT  D$; "OPEN INVENTORY,L15"
130  PRINT  "ENTER DATA AS:";
140  PRINT  "NO., UNIT PRICE, QUANTITY"
150  PRINT  "TYPE 0, 0,0 WHEN FINISHED."
160  INPUT  P, UP, QT
170  IF  P =0 THEN 220
180  PRINT D$; "WRITE INVENTORY, R", P
190  PRINT UP;", "; QT
200  PRINT D$
210  GOTO 130
220  PRINT D$;"CLOSE INVENTORY"
999  END
```

这个程序将建立一个名为“INVENTORY”的随机文件，它的记录长度为15个字节，记录中包括单价、数量、货物的编号（即文件中记录编号）。记录中，单价应不超过五个字节，数量最多占四个字节，它们之间的逗号占一个字节，最后有一个“RETURN”占去一个字节，加起来共11个字节，剩下的四个字节，则留作日后扩展用。

读者可利用本章后附的实习思考题17的数据，建立一个

"INVENTORY"文件。

该程序的第120句打开文件，后面的L15代表记录长度；第160句将输入数据存入编号为P的记录中；第170句判断输入是否结束。程序中没有对输入的数据进行校验，而是直接写入磁盘文件中。

2. 随机文件中指定资料的读出

读出随机文件中指定资料的基本方法是：

(1)打开随机文件。注意记录长度必须和它建立时的相同；

(2)用READ及INPUT命令读取相应的记录；

(3)关闭文件。

【例9】读出上例中"INVENTORY"文件中的资料。

```
100  REM  RETRIEVE INVENTORY
110  D$ =CHR$ ( 4 )
120  PRINT  "NO.","UNIT  PRICE";"
     QUANTITY"
130  PRINT  D$;"OPEN INVENTORY, L15"
140  FOR R=1 TO 13
150  PRINT D$;"READ INVENTORY, R"; R
160  INPUT UP, QT
170  PRINT  D$
180  PRINT  R, UP, QT
190  NEXT R
200  PRINT D$;"CLOSE INVENTORY"
999  END
```

本程序将读出文件"INVENTORY"中的1～13号记录。

3. 随机文件的更新

随机文件更新的方法较多。这里主要通过下面例子说明其基本方法。

【例10】这个程序用于更新“INVENTORY”文件。按照给出的货物编号（即记录号），程序可以提供：

（1）修改单价及数量（修改一个记录）；

（2）增加或减少数量。

```
100 REM UPDATE INVENTORY
110 D$ =CHR$(4)
120 HOME
130 PRINT D$;"OPEN INVENTORY,L15"
140 PRINT "GIVE ME THE RECORD NO.,
    TYPE 0 TO EXIT."
150 INPUT R
160 IF R=0 THEN 410
170 PRINT "ENTER YOUR SELECTION:"
180 PRINT "1) CHANG THE RECORD"
190 PRINT "2) RECEIPT OR SENDOUT"
200 PRINT "3) QUIT"
210 PRINT "?";: GET T: PRINT T
220 PRINT D$;"READ INVENTORY,R";R
230 INPUT UP,QT
240 PRINT D$
250 PRINT "RECORD","UNIT PRICE";"
    QUANTITY"
260 PRINT R,UP,QT
270 INPUT "RIGHT (Y/N)?:";T$
```

```
280 IF T$ = "N" THEN 140
290 IF T$ < > "Y" THEN PRINT
    CHR$ (7); "ERROR": GOTO 270
300 ON T GOTO 310, 340, 410
310 PRINT "CHANGE: TYPE THE NEW
    UNIT PRICE, QUANTITY"
320 INPUT UP, QT
330 GOTO 370
340 PRINT "RECEIPT OR NEGATIVE
    VAULE TO SENDOUT"
350 INPUT TA
360 QT = QT + TA
370 PRINT D$;"WRITE INVENTORY, R";R
380 PRINT UP;",";QT
390 PRINT D$: PRINT "DONE!"
400 GOTO 140
410 PRINT D$;"CLOSE INVENTORY"
999 END
```

该程序部分语句的含义是：

第130行打开文件"INVENTORY"；

第150句从键盘上获得记录号（即货物编号）；

第180～210句作选择处理，提示本程序的功能：打入1——修改记录；打入2——接收货物或发出货物，输入负值为发出；打入3——结束；

第220～290句从文件中读出指定的记录，并显示出来供核实；

第310～320句修改记录，输入新的值；

第340～360句，接收和发出货物处理。

关于文本文件命令的更详细说明，请参考DOS系统使用手册，由于篇幅所限，本节不再赘述。

第六节　DOS系统盘中的应用程序

在APPLE Ⅱ微型计算机的DOS3.3系统磁盘中，除DOS系统外还附有一系列的程序。这些程序大致可分为三类：第一类是系统的应用程序，如FID、COPY及MUFFIN等；第二类是示范程序，如COLOR DEMO、MAKE TEXT和EXEC DEMO等；第三类是游戏程序，有ANIMAL、LITTLE BRICK OUT等。

下面介绍DOS系统盘中的应用程序及其使用方法，其中COPY及COPYA程序在本章第二节已作了介绍。

1. CHAIN

CHAIN程序在链接两条浮点BASIC程序时使用。它可以将第一条程序的变量、字符串保留下来，并传递到第二条程序中。这一功能对用户使用内存不能容纳的大程序，是非常有用的。CHAIN程序可以将用户程序分为几个部分，动态地链接使用。

CHAIN程序的使用方法举例如下：

```
60000 PRINT CHR$(4); "BLOAD CHAIN,A520"
60010 CALL 520"PART TWO"
```

这段程序必须连在一起使用。第60000句将CHAIN程序载入内存，第60010句对文件名为PART TWO的程序进行链接。注意：CALL地址与文件名引号之间不能有空格。

2．BOOT 13

这条程序允许APPLE Ⅱ机调用13个扇区的磁盘（因为DOS3.3以前的版本格式都是每磁道13个扇区）。可以用BRUN命令执行这条程序。

3．FID

DOS系统盘中的FID程序可提供以下的功能：

（1）复制各种文件；

（2）查磁盘空间（包括已用的和未用的）的使用状况；

（3）删除、上锁和解锁、检验（VERIFY）及列出文件名清单等。

FID程序可以用BRUN FID调用，进入选单。

文件的复制方法如下：

用户常常需要复制文件，尤其是复制文本文件。这时，用户可以从键盘上打入1，机器即会提示用户回答源盘和复制盘所在的接口板及驱动器编写，回答需要复制的文件名。这里的文件名可以使用特指的，也可以使用泛指的。在指定文件名时，可以使用一个通配字符“=”。如打入“F=”，则表示复制以字母F开头的所有文件；打入“=·DAT”，则表示复制以“·DAT”结尾的所有文件；打入“=”，则表示复制源盘上的全部文件。

这时，FID程序将提问：

DO YOU WANT PROMPTING?（需要提示吗?）

如打入“N”（不需要），则FID程序就会将源盘上的全部程序抄到复制盘上。

如打入“Y”（需要提示），则FID程序会将源盘文件目录中的文件名逐个显示在屏幕上，然后等待用户回答。若用户打入“Y”并按下RETURN键，机器就开始复制文件。

若用户打入“N”并按下RETURN键，则 FID 会跳过这个文件，再从目录中取下一个文件名，直到磁盘上最后一个文件为止。

如果要复制的文件名在复制盘上已存在，则FID会给出信息。用户可按下述三种方式进行选择：

（1）按输入的新文件名复制；

（2）按下RETURN键，将旧的文件置换；

（3）按下“CTRL-C”键，再按RETURN键，取消复制要求。

检查磁盘空间的使用状况的方法如下：

用户从键盘上打入3，在选择好接口板及驱动器编号后，屏幕便会显示出该磁盘的使用状况（已用扇区数及未用扇区数）。如果用户要重新查另一个驱动器中磁盘的使用状况，则应先打入7，以取消原来的驱动器及接口值，重新定义新的值，否则，FID程序仍从原来的驱动器上查寻。

4. MUFFIN

MUFFIN程序的功能是：将13个扇区磁盘上的文件抄到16个扇区磁盘上。用户可用BRUN MUFFIN命令将该程序调出执行。其操作过程和FID程序基本相同。

5. RENUMBER

RENUMBER程序有两项功能，其一为换行号，可以更换浮点BASIC程序的全部或部分行号；其二为合并程序，可以将两条程序合并为一条。在系统盘中，还有一条RENUMBER INSTRUCTIONS 程序，它详细介绍了RENUMBER程序的使用方法。

RENUMBER程序可以用RUN RENUMBER命令调出执行，该程序的两项功能可分别用下述两种命令实现。

（1）换行号命令

& 〔F10〕 〔,I10〕 〔,S0〕 〔,E63999〕

其中：F——开始行号；

I——行号间隔；

S——需要换行号的起始行号；

E——需要进行换行号的结束行号。

运行了RENUMBER程序后，将需要换行号的程序用LOAD命令载入内存，然后用换行号命令更换程序中的行号。命令字母后面的数字是省缺值。例如：

& FIRST 10,INC 10,START 10,END 5000

& F10,I10,S10,E5000（或 & E5000）

上面这两个命令的含义是相同的，它令RENUMBER程序将内存中的浮点BASIC程序第10行至第5000行的行号，更换为从行号10开始，间隔为10的格式。

（2）合并程序

将要合并的两条程序的其中一条调入内存，打入&H命令（Hoid），先将该程序锁存起来，然后调入第二条程序，打入命令：&M（Merge），两条程序就合二为一了。语句按行号升序排列，如果两条程序中有相同的行号，则第一条程序的语句排在后面。

实习与思考题

1．试从系统磁盘中调出DOS系统。

2．用INIT HELLO命令对一个磁盘初始化，要求HELLO程序可以在屏幕上列出该磁盘上文件的目录。

3．如果要将内存中的一条BASIC程序存入磁盘（DOS

系统已调入内存，文件名用“TASK”），正确的命令应是以下的哪一条？

（1）SAVE

（2）TASK SAVE

（3）INIT TASK

（4）SAVE TASK

（5）WRITE TASK

4．试用COPYA程序，复制一个磁盘。

5．试用FID程序：

（1）查磁盘空间的使用情况；

（2）将一个文件从一个磁盘上复制到另一个磁盘上。

6．如果不用COPY或COPYA程序，而用系统盘上的其他程序，可以将一个磁盘（DOS3.3格式）上的内容全部抄到另一个磁盘上吗？

7．试将＄03D0～＄03FF地址中的内容（DOS向量表）存入磁盘。

8．以下几段程序的功能是否相同（其中D＄＝CHR＄(4)）？

(1)
```
100 PRINT D$;"BLOAD SORT.OBJ"
110 PRINT D$;"BRUN SORT"
```

(2)
```
100 PRINT D$;"BRUN SORT"
110 PRINT D$;"BLOAD SORT.OBJ"
```

(3)
```
100 PRINT CHR$(4)+"BLOAD SORT.
    OBJ"+CHR$(13)+CHR$(4)+"BRUN
    SORT"
```

(4)
```
100 PRINT D$;"BRUN SORT";D$;
```

"BLOAD SORT•OBJ"

9. 将本章例5 MAKE PHYSICS程序输入计算机，试输入以下资料，建立一个"PHYSICS"的文本文件。

姓名	学号	成绩
WANG PING	10	87
FANG LING	9	60
CHEN MIN	24	89
WANG TING BIAO	34	90
SU SHI KAI	1	95
YAO LING LING	13	99
LIN JING DONG	14	112
HONG XIAO BAO	16	114
QIAN HE DUO	43	49
LIU BO MIN	22	120

10. 用例题 RETRIEVE PHYSICS 及 FIND A RECORD检验PHYSICS文件。

11. 运行DOS MASTER中的MAKE TEXT及RETRIEVE TEXT程序。

12. 将MAKE EXEC例题输入计算机并运行之，用它建立的DEMO文件生成APPEND PHYSICS程序，并用这条程序将下列资料加在PHYSICS文件后面。

姓名	学号	成绩
ZHAO YANG	11	115
LI XIAO PIN	31	68
GAO QI	28	120

13. 利用前面所建立的 PHYSICS 文件的数据，写一条程序能打印出学生成绩报表，要求报表按成绩降序排列。

14．对于一个文本文件，要想了解其内容，最简单的方法是先进入MON状态，然后用EXEC命令检查。如果有一个文件，按上述步骤执行后，机器马上回到命令状态。请问文件可能是什么样的文件。

15．试建立一个EXEC文件，它可以把程序EXEC DEMO从DOS MASTER中调入内存，然后再存入另一个磁盘（用两台驱动器）。

16．了解系统中的EXEC DEMO程序。

17．运行本章例8的"MAKE INVENTORY"程序，用给出的资料建立库存文件"INVENTORY"，然后用RETRIEVE INVENTORY检验库存。

编号	单价	数量
26	5.25	110
108	24.95	70
2	4.66	77
17	19.95	255
189	4.6	10
24	2.32	400
49	10.5	90
7	9.95	340
10	14	200
82	1.96	420
12	3.1	226
152	4.85	147
86	7	110

18．用本章例10的UPDATE INVENTORY，修改"INVENTORY"文件。

19．本章例10的UPDATE　INVENTORY程序是采用记录号（编号）来进行检索的，试编出采用货物单价进行检索及更新的程序。

20．本章例10的UPDATE　INVENTORY程序不能对新的（文件中不存在的）货物进行处理，试增加某些语句，使该程序具有这项功能。

Retro Workshop 果粉工作室

第十章　打印机的使用方法

一、打印机简介

打印机是计算机的一种有效的输出设备，它可以打印输出计算机的程序清单、程序运行结果以及报表等等。

在APPLE Ⅱ微型计算机上使用打印机，必须配备一块打印机接口板（或称打印卡），通过它来控制打印机。

一般微型计算机使用的打印机可按其打印方式分为点阵式、喷墨式及字符质量式三种。点阵式打印机又可分为撞击式及热敏式两种。

打印机还可以按其数据传输方式，分为并行式和串行式两种。点阵式打印机多数采用并行式。

通常，在APPLE Ⅱ微型计算机上连用的打印机都是点阵撞击式的。这种打印机通过其打印头上的撞针（9～24根），撞击打字色带，而在纸上打印出英文、符号或图形。针数越多打印分辨率越高。其打印速度一般为每秒80字符至160字符，最高可达400字符。打印宽度一般为80列或132列。

EPSON系列打印机主要有80列的MX-80、RX-80、FX-80三种型号及132列的MX-100及FX-100两种型号。与其兼容的机型有：STAR Gemini 10X和15X；STAR Delta 10；Super 5 CP-80等。

本章将介绍EPSON系列点阵撞击式打印机或类型兼容的打印机及其接口板的使用方法。这两种并行打印机的接口板，分别为EPSON EP12并行接口板及GRAPPLER+并

行接口板。

打印机接口板，一般插在主机的#1扩充插座上。在以下的介绍中，n代表打印机接口板所在的扩充插座编号。

注意：在使用打印机之前，应该首先查阅所用的打印机说明书，以确认该打印机上的控制开关（DIP SWITCH）处于打印卡要求的位置上。这是因为有些打印机在打印英文或图形（中文）时，必须重新置定控制开关的位置。

用打印机输出中文的方法将在第十一章介绍。

二、EPSON打印机及其接口板的使用方法

1．在浮点BASIC或整数BASIC中使用打印机

在这两种BASIC语言中，用户都可以用PR#n命令调用打印机、如果打印机接口板插在#1扩充接口插座中，在命令状态下，可以打入命令：

PR#1

启动打印机接口板，使所有显示信息除送屏幕显示外，均送打印机输出。

在程序执行过程中调用打印机，要分两种情况处理：

（1）当不用DOS系统时，可在PR#n命令前加上行号。例如：

100 PR#1

这个语句被执行后，以后的PRINT语句中的内容除显示外，都输往打印机打印。

（2）当使用DOS系统时，则PR#n应作为DOS命令使用。例如：

100 PRINT D$；"PR#1"

这里的PR#n命令和不用DOS时的PR#n命令是有区别的（其详细说明请参阅DOS参考手册）。

如要停止打印，可以用PR＃0命令，使PRINT命令的内容只显示在屏幕上。在DOS系统中应该用：

100 PRINT　CHR$（4）；"PR＃0"

2．在监控状态下使用打印机

在监控状态下使用打印机可分为两种情况处理：

（1）进入监控状态时，可以用DOS命令：

PR＃n；CALL—151

（2）内存中有DOS系统时，可以用DOS命令：

＊PR＃n

以上命令被执行后，其后屏幕上显示的内容即可同时在打印机上输出。

若要停止打印而无DOS系统时，可按RESET键；有DOS时可打入PR＃0命令。

3．在汉字状态下使用打印机

请参考第十一章。

4．在CP/M系统状态下使用打印机

要在CP/M系统状态下使用EPSON打印机接口板及打印机，必须将打印机接口板插在＃1插座上，而且要对系统作一点修改，方法如下：

从CP/M系统盘（MASTER）中调出CP/M系统，进入系统后，屏幕上出现：

A>■

这时，打入DDT（关于DDT可参考CP/M使用手册）。

A>DDT

DDT　VER　2.2

—■

进入DDT后，可以进行如下修改：

（1）对于56K CP/M系统，可以打入

```
—SDD2F
DD2F  3E  ■
```

将3E 改为31，打入31，按RETURN

```
DD2F  3E  31
DD30  DD  ■
```

打入CTRL－C，退回CP/M系统状态：

```
DD30  DD  ∧C
A>
```

至此修改完毕。

（2）对于44K CP/M系统，只需将上面所有的DD2F用AD2F替换即可。

系统经过修改后，即可使用EPSON打印机。

若按CTRL－P组合键，系统就会将显示在控制台上的内容复制到当前定义的LST，即在打印机上输出；若再按CTRL－P，则取消LST，即停止在打印机上输出。

在MBASIC、GBASIC、FORTRAN-80、COBOL-80等CP/M高级语言中用打印机输出的方法，可参考有关手册。

5．设定打印格式

APPLE Ⅱ型计算机的显示格式为每行40个字符，在浮点BASIC或整数BASIC状态中列程序清单时，每行只显示33个字符，而打印机的打印宽度通常为每行80字符。由于PR＃n命令是将屏幕内容送打印机打印的，因此80个字符的打印机只能印33～40个字符，这就太浪费了。

地址＄678＋＄0n（1656＋n），可用于设置打印机每行输出的字符数目（n为打印机接口板所在接口编号）。如果这个地址单元的值大于40，字符只在打印机上打印，不在屏

幕上显示。当每行输出了设定数目的字符后，会自动换行，例如：

POKE 1657，80（ $678＝1656）

这个命令将打印机每行输出的字符数目定为80个。（最多132个）

6．打印高分辨率图形

高分辨率图形可以用"CTRL-Q"命令要打印机打出来。方法是：

PR＃1

"CTRL-Q" （按CTRL＋Q）

或 100 PRINT D$；"PR＃1"

110 PRINT CHR$（17） （注：CHR$（17）即"CTRL-Q"）

这时打印机就可打印出内存中的高分辨率图形。打印的格式是由地址单元$778＋$0n（n为打印机接口板所在的接口编号）的值确定的。下面是该地址单元中每一位被置位为"1"时的含义：

高分辨率图形打印的选择码

位	7	6	5	4	3	2	1	0
	行单位	放大	反转	异或	或	与	第二页	第一页
值	128	64	32	16	8	4	2	1

地址：$778＋$0n或1912＋n；

位7——以行为单位打印高分辨率图形；

位6——采用扩充格式打印；

位5——将信息反转后打印（当位5为0时，高分辨率图形在屏幕上的一个亮点在打印机上输出为黑点）；

位4——打印出两页高分辨率图形区中的不重合图形点；

位3——将两页高分辨率图形重叠打印出来；

位2——打印出两页高分辨率图形区中的重合图形点；

位1——打印第二页高分辨率图形；

位0——打印第一页高分辨率图形。

【例1】

```
100  POKE  1913, 66
110  PRINT  CHR$(4); "PR#1"
120  PRINT  CHR$(17)
130  PRINT  CHR$(4); "PR#0"
```

这段程序执行的结果是在打印机上用扩充打印格式打出第二页高分辨率图形。打印机接口板插在#1接口上。地址为1913，其值66，即$42。这样，位6及位1被置为1，其余为零。

若内存中没有DOS系统，则第110句及130句要直接用"PR#1"和"PR#0"，而不能用PRINT "CTRL-D"的方式。

在命令状态下，可以打入：

```
POKE  1913, 3
PR#1
"CTRL-Q"   (按CTRL+Q)
```

如果按照上述步骤操作，则可以命令打印机将两页高分辨率图形并排打印出来，地址单元1913中位1及位0被置位。

注意：某些80列的打印机（如EPSON MX-80、FX-

80及STAR Gemini 10X、STAR、Delta 10打印机）受到宽度限制，不能将上述两例的图形全部印出。

利用POKE 1144+n，76命令可以使一些打印机打印出双倍密度的高分辨率图形。用POKE 1144+n，75命令可变回单密度打印格式。

三、GRAPPLER+接口板的使用方法

除了EPSON打印机接口板外，常用的打印机接口板还有APPLE PARALLEL、GRAPPLER及GRAPPLER+。其中GRAPPLER+的功能最强，使用比较广泛，并且可以和多种打印机连接。下面介绍它的主要控制命令，供参考。APPLE PARALLEL及GRAPPLER打印机接口板的大多数命令是和GRAPPLER+兼容的。GRAPPLER+打印机接口板的输出电缆是26芯的，EPSON打印机接口板是20芯的。

1. 控制文本打印格式的命令

（1）PR#n

功能：启动打印机接口板，调用打印机输出。所有控制打印机的命令必须在执行了这个命令以后使用，否则打印机接口板不接收。

【例2】PRINT CHR$(4)；"PR#n"

（2）PR#0

功能：关闭打印机接口板。

（3）CTRL-IA

功能：在"RETURN"（回车）命令之后加上"LINE-FEED"（换行）命令，前进一行。BASIC在回车之后不送出换行命令，GRAPPLER+接口板会自动送出一个换行命令。

【例3】PRINT CHR$(9); "A"

(4)CTRL-IK

功能：在回车之后不加换行命令。

【例4】PRINT CHR$(9); "K"

(5)CTRL-InN

功能：将行的长度定义为n个字符(左边边界算起)。执行这个命令之后，输出的信息仅送打印机打印，不送屏幕显示。当打印完n个字符后，打印机接口板会自动送出一个回车命令。如果n=0，则接口板只在APPLE Ⅱ机上接收到回车命令之后才送出回车命令。当开列程序清单时，n被自动定义为40，用这个命令可以改变n的值。

【例5】PRINT CHR$(9); "80N"

(6)CTRL-II

功能：将输出信息同时向屏幕及打印机传送。这个命令可以和CTRL-InN命令一起使用，但在开列程序清单时无效。

【例6】PRINT CHR$(9); "I"

(7)CTRL-InL

功能：设定打印机左边边界在第n列。

【例7】PRINT CHR$(9); "10L"

(8)CTRL-InR

功能：设定打印机右边边界至第n列。当n=0时，撤消此功能。

【例8】PRINT CHR$(9); "75R"

(9)CTRL-IS

功能：将英文(TEXT)屏幕复制到打印机上输出。此命令只能复制40列的屏幕，不能复制80列或更宽的屏幕。

【例9】PRINT CHR$(9)；"S"

(10)CTRL-I2S

功能：把两页文本屏幕区的内容并列输送到打印机上以80列格式输出。该命令只对40列的屏幕显示有效。

【例10】PRINT CHR$(9)；"2S"

(11)CTRL-InP

功能：设定每页长度为n行，当打印完n行后自动送出6个换行命令分页。打印纸每页的标准行数为66行，n=0时不分页。

【例11】PRINT CHR$(9)；"60P"

2. 控制打印图形的命令

GRAPPLE+打印卡可以用CTRL-IG命令后加选择子命令的方法，将高分辨率图形送打印机打印。如果不加任何选择子命令，打印机将水平地印出第一页高分辨率图形，屏幕上的亮点在打印机上输出为黑点。

下面介绍八种选择子命令。

(1) 2

功能：打印第2页高分辨率图形。

【例12】PRINT CHR$(9)；"G2"

(2)S

功能：并列打印第1及第2页高分辨率图形。这个子命令不能与旋转及双倍宽打印命令联合使用。一些80列的打印机(如EPSON MX-80、EPSON RX-80；OKIDATA 82A)，在使用这个子命令时不够宽度。

【例13】PRINT CHR$(9)；"GS"

(3)M

功能：打印高分辨率图形及4行文本显示。使用这个子

命令时，旋转、双倍宽打印及左边界命令无效。

【例14】PRINT　CHR$(9);"GM"

(4)D

功能：用双倍宽度打印高分辨率图形。一般80列打印机（如EPSON MX-80EPSON RX-80，OKIDATA 82A、OKIDATA 92A），在使用这个子命令输出时不够宽度。在这种情况下，必须和“R”子命令联合使用。

（5）　E

功能：用加强格式打印。EPSON系列打印机、NEC 8203、C.Iton 8510、DMP 85及APPLE Dot Matrix等打印机，都具备此功能。

【例15】PRINT　CHR$（9）；"GE"

（6）　I

功能：将高分辨率图形反转后再打印。

【例16】PRINT　CHR$（9）；"GI"

（7）　L

功能：将高分辨率图形在预先设定的左边界上开始打印。注意：有些打印机不具备这项功能，或受打印格式影响。

【例17】PRINT　CHR$（9）；"GL"

（8）　R

功能：将高分辨率图形顺时针旋转90°再打印。一些80列的打印机，在用双倍宽打印时需要这项功能。

3．使用GRAPPLER＋打印机接口板的实例

（1）输入控制命令

用户若要打印机以80列方式列出程序清单，并且按每页60行自动分页，可以用以下方式输入控制命令：在命令状态下打入命令PR＃1，启动打印机接口板，然后打入命令{CTRL-

I}80N{CTRL-I}60P，并按下RETURN键（这时打印机上会输出：？SYNTAX ERROR。这是正常的，因为BASIC不认识这命令，但打印机接口板会接收它）。这时打入LIST命令，用户就可以得到按指定格式打印出来的程序清单了。

注意：1）命令{CTRL-I}80N{CTRL-I}60P必须用在PR＃1之后，否则打印机接口板不会接收；2）不能在命令中（指CTRL-I后面）加入空格或其它字符，但在两个命令之间加入则无妨碍。

以上命令可以在程序中使用，打入

```
10  PRINT  CHR$(4); "PR#1"
20  PRINT  CHR$(9); "80N"
30  PRINT  CHR$(9); "60P"
```

然后运行这段程序就可以了，这样还可避免出现？SYNTAX ERROR的信息。

（2）打印高分辨率图形

下面的程序可以使打印机采用双倍宽及旋转90°的方式，打印出反转的第2页高分辨率图形。子命令选择可以用不同的排列，只要以CTRL-IG开头，后面加上回车键就行了。

```
10  PRINT  CHR$(4)"PR#1"
20  PRINT  CHR$(9); "GD2R"
```

(3)GRAPPLER＋打印机接口板可以直接用于CP/M、Pascal及APPLE机的汉字卡。

Retro Workshop
果粉工作室

第十一章　汉字卡的使用方法

本章所介绍的汉字卡，以台湾佳佳公司出品的C-PLUS Ⅱ汉卡为例，也适用于其它同类产品。

一、汉字卡简介

使用汉字卡时，卡上应具备如下硬件：

CPU:Z-80A

RAM:2K字节（6116）（APPLE机至少需要48K）

ROM:40K字库，12K监控

I/O:8255

汉字卡具有如下功能：

（1）采用仓颉字母输入法则组成繁体汉字；

（2）字库可产生23000个组合汉字，其中有效汉字约6000个；

（3）只能用APPLE Ⅱ或APPLE Ⅱe的FP BASIC（APPLESOFT）语言，在第二页高分辨率图像区间作屏幕显示。显示方式:中文为17字/行，10行/帧；英文及字符为34字/行，10行/帧。中文字由16×16点阵组成。

二、汉字卡的调用

1. 将汉字卡插在主机的1～7中的任一个扩展接口上，一般插在第3号扩展接口，即从左边数起第四个。注意:在插入汉字卡前应先将电源关闭。

2. 开机后（如果有磁盘机的，可以调入DOS系统），进入FP BASIC（APPLESOFT）状态。这时，可用PR＃n

命令调用汉字卡，如：

PR＃3

即可进入汉字卡显示状态。

3．汉字卡的中文输入是采用英文字母A～Y键，每个键符代表中文字的一个部首边旁，即所谓仓颉输入法。关于仓颉输入法，可以参考有关书籍。

4．英文字母和中文字母输入状态是通过CTRL-L切换的。即：

英文输入状态→CTRL-L→中文输入状态→CTRL-L→英文输入状态

进入中文输入方式后，屏幕左下角会出现"中文"两字，组字时所采用的部首也会显示在"中文"两字的旁边。部首码输完后，要按一下空格键，中文字才会在屏幕上组成。如果输入完毕，应先退出中文输入状态，再按RETURN键。

如果用户的APPLE机有输入小写字母的功能，那么按下小写键后，即可进入中文输入状态，而且在中文输入完毕后无须退回英文状态再按RETURN。

注意：

（1）由于小写键与CTRL-L具有相同功能，两者同时作用时，等于按了两次CTRL-L组合键或小写键，输入状态不会改变。

（2）左下角"中文"两字及组字时的部首的显示与否，由CTRL-D决定。CTRL-D也是采用切换方式的。

（3）有一种较简易的方式可以用于退出中文输入状态，那就是当打完中文后，先按"→"键或空格键，再按"←"键，这比使用"CTRL-L"方便一些。

三、仓颉字母汉字组码输入简则

1．取码规则

（1）取码顺序——由外而内，由上而下，由左而右；

（2）连体字取码——依序取首、次、三、尾四码；

（3）字首取码——限取一至二码，超过则取首、尾二码；

（4）字身取码——取一至三码，依序取首、次、尾三码。

2．取码方法

（1）取码精简——有多种取码可能时，以码数少为正确；

（2）取码完整——取码数若相等，先取的字形必须完整；

（3）字形特征——在前两原则下，不能放弃原字形的特征。

3．省略原则

（1）部分省略——超过取码数，不取的码自动省略；

（2）包含省略——必须省略时，先省略包含在内的字形。

【例1】

中L	理MGWG	研MRMT
山U	系HVIF	究JCKU
大K	微HOUUK	所HSHML
学HBND	電MBWU	
物HQPHH	子ND	

中文字母表

哲理类	笔划类
A：日、⺫	H：竹、丿、厂（斜）
B：月、冂、冖、⺫、⺈	I：戈、丶、厶、广（點）
C：金、丷、八、儿	J：十、亠（交）
D：木、寸、廾	K：大、乂（叉）
E：水、又、氵	L：中、串、丨（縱）
F：火、小、灬、⺌	M：一、工、厂（横）
G：土、士	N：弓、亅、⺂、⺈（钩）

人体类	字形类
O：人、く、㇏	S：尸、コ、㇆、匚、E（侧）
P：心、勹、匕、七、忄、⺗	T：廿、艹、卄、丷、卯（並）
Q：手、扌、扌、龵、牛	U：山、凵、丨、屮、乚（仰）
R：口	V：女、く、㐅、乚（组）
	W：田、口、日（方）
	Y：卜、亠、⺊、二、冫、礻
	X：重/难键。

四、软件功能

（1）在汉字卡状态中，以下的 FP BASIC（APPLE-SOFT）命令不能使用：

TEXT；VTAB；HTAB；HOME；TAB；POS；SPC；FLASH；INVERSE；NORMAL；GR；COLOR；PLOT；HLIN；VLIN；SCRN；HGR

除此之外，其余FP BASIC命令可照样使用。

（2）用于控制显示的一些CALL命令，如：CALL-958和CALL-936不能使用。

另外，TEXT WINDON（开窗口）命令也不能使用。

（3）在FP BASICS 的ESC编辑指令中，除ESC E及ESC F外，其余的指令均可使用。

（4）由于汉字卡占用了$9000～95FF的地址，因此，在使用DOS系统时，不能改变MAXFILES的值（原值为MAXFILES3）。

五、子程序的调用

子程序可以用"CALL"命令调用，子程序的功能及其入口地址如下表。

名　称	功　　　能	入口地址
HM	清屏并将光标移到左上角，相当于“HOME”	BS+21
CC	清除前面的 PRINT 命令留下的光标	BS+24
CE	清除至行末，相当于ESC　F	BS+36
CL	清除一行	BS+39
CF	自光标起清除至帧末，相当于ESC　F	BS+42
BS	表示49152+256*CN	
CN	表示汉字卡所在的扩充接口号码	
TX	从汉字卡返回英文显示（TEXT）状态	64098

如果汉字卡插在#3扩充接口上，要清除屏幕上的内容，可以打入：

CALL　49941

或10　CALL　49941

其中49941 = BS + 21 = 49152 + 256×3 + 21

六、调用软开关的POKE命令

在汉字卡状态下，调用软开关的POKE命令及其作用如下表所示。

命　　令	作　　用
POKE　214，H	相当于HTAB功能，H值为0～33
POKE　215，V	相当于VTAB命令，V值为0～9
POKE　253，0	显示中文时，屏幕左下角不显示中文字码
POKE　253，255	显示中文时，屏幕左下角显示中文字码

七、使用汉字卡时内存的分配

使用汉字卡时要求主机有48K的内存。其中，汉字卡显示占用了第二页高分辨率图像（HGR2）地址$4000～

$ 5FFF；DOS 系统占用 $ 9600～$ BFFF（三个文件缓冲区MAXFILES 3)；汉字卡本身占用地址 $ 9000～$ 95FF。因此，内存的用户使用汉字卡时，必须 HIMEM：= $ 9000。如下表所示。

	$ BFFF
DOS 系统	
	$ 9600
中文板占用	
	$ 9000
用户 RAM	
	$ 6000
中文板显示(HGR2)	
	$ 4000
用户 RAM	
	$ 0800
系统 RAM	
	$ 0000

从上表可见，内存的用户RAM被分为不连续的两部分，一部分的地址为 $ 0800～$ 3FFF 共14K 字节，另一部分地址为 $ 6000～$ 9000 共 12K 字节。由于 FP BASIC 是从 $ 0800开始存放的，当程序长度超过14K 字节时，就会和显示区间发生冲突。$ 6000～$ 9000的用户区间有12K 字节，一般用于存放变量、字符串及数组，或机器语言子程序。

由于汉字卡占用了 $ 9000～$ 95FF 的内存，DOS 操作系统占用 $ 9600～$ BFFF，因此，如果改变 MAXFILES 的值，将会引起不良的结果，而且可能是灾难性的结果。由于MAXFILES的自然值为MAXFILES 3，故用汉字卡时，最多只能使用（OPEN）三个文件。

八、使用打印机输出中文

汉字卡必须与EPSON打印机接口板或其它类似产品(如GRAPPLER+接口板)连用。由于汉字卡已占去一个扩展接口，因此在汉字卡状态下调用打印机输出时，不能用PR#n命令，而只能通过POKE命令调用软开关进行。在下表中，CN代表汉字卡所在的接口号码，PT代表打印机接口板(EPSON)所在的接口号码。

在汉字卡状态下，调用打印机的POKE命令及其功能如下表所示。

命　　令	功　　能
POKE 1400+CN, PT	调用打印机(相当于PR#PT)
POKE 1400+CN, 0	撤消打印机输出(相当于PR#0)
POKE 1656+CN, 0	中文大字横印
POKE 1656+CN, 1	中文大字直印
POKE 1656+CN, 2	中文小字横印
POKE 1656+CN, 3	中文小字直印
POKE 1912+CN, N	定义英文字间隔为N点，中文字间隔为2N点，一般N=0
POKE 1784+CN, N	定义行与行之间的距离为N点，一般N=4
POKE 2040+CN, N	表示打印机每行打出的字数，一般N=50

【例2】下面是一条在中文状态下的程序，它能在打印机上印出中文字的几种不同字体。

```
10  REM  中文打印格式
20  CN=8
30  BS=49152+256*CN
```

```
40  HM=BS+21
50  CALL HM:REM HOME
60  PRINT "  "
70  PRINT "已开打印机(Y/N)? :";
80  GET K$:PRINT T$
90  IF K$="N"THEN PRINT "请开打印机":
    GOTO 70
100 IF K$< >"Y" THEN PRINT CHR$(7):
    GOTO 50
110 POKE 1400+CN,1: REM 调用打印机
120 PRINT"中文打印格式: "
130 POKE 1656+CN,0
140 PRINT "大字横印"
150 POKE 1656+CN,1
160 PRINT "大字直印"
170 POKE 1656+CN,2
180 PRINT "小字横印"
190 POKE 1656+CN,3
200 PRINT "小字直印"
210 POKE 1656+CN,0: REM 大字直印
220 POKE 1400+CN,0: REM PR#0
230 END
```

该程序执行结果如下:

中文打印格式:

大字横印

大字直印

小字横印

小字横印

在中文状态下用打印机打印时，计算机自动定为："大字横印"格式。

将这两种状态下的程序相比较，读者可以发现，中文字在机器中实际上是用小写的英文字母代码组成的。这就说明了为什么用户可以按下小写键输入中文，而不需要按"CTRL-L"。另外，读者还会注意到：230句的END语句在中文状态下的输出变成了E；LEFT $ 函数在中文状态下输出是L。但这并不说明用户在输入程序时可以省略。

将上述中文状态下的程序，改为英文状态下的程序，如下：

```
10  REM  l  yk  qmn  hpsl  dher  ipm
20  CN=3
30  BS=49152+256*CN
40  HM=BS+21
50  CALL  HM: REM  HOME
60  PRINT  "  "
70  PRINT  "xsu mt qmn hpsl dhu  (Y/N)?:";
80  GET  K$: PRINT K$
90  IF  K$="N"  THEN  PRINT  "yrqmb mt
    qmn hpsl dhu": GOTO  70
100  IF  K$<  >"Y"  THEN  PRINT  CHR$
     (7): GOTO  50
110  POKE  1400+CN, 1: REM  yrbgr bq qmn
     dhu
```

```
120  PRINT "l yk qmn hpsl dher ipm :"
130  POKE  1656+CN, 0
140  PRINT  "k jnd dtwc hpsl "
150  POKE  1656+CN,1
160  PRINT "k jnd jbmm hpsl "
170  POKE  1656+CN,2
180  PRINT "nc jnd dtwc hpsl "
190  POKE  1656+CN,3
200  PRINT "nc jnd jbmm hpsl"
210  POKE  1656+CN,0: REM k jnd jbmm hpsl
220  POKE  1400+CN,0: REM  PR#0
230  END
```

【例 3】下面这段小程序是在英文（TEXT）状态下列出的，其作用是利用打印机打印出较宽的行。80 列的打印机，每行可以打出120个英文小字或60个小汉字，这个数目在第140句设定，其默认值为50，同时请注意A$的内容。

```
100  CN=3
110  A$="a b c d e f g h i j k l m n o p q r
     s t u v w y"
120  A$=A$+A$+A$
130  POKE  1656+CN,2
140  POKE  2040+CN,120
150  POKE  1400+CN,1
160  PRINT  A$
170  POKE  1656+CN,0
180  POKE  1400+CN,0
190  POKE  2040+CN,50
```

```
200 END
```

当用户要使用汉字卡时，是否可以从键盘上直接打入“PR＃n”命令，使计算机从英文TEXT状态转入中文显示状态呢？如果机器中的汉字卡已经被调用过（即曾被“冷起动”，那是可以的。但如果用户想在程序开始时或在运行过程中进入中文显示状态，而不管汉字卡是否曾被冷起动，或者想在开机时通过机器的“欢迎”程序进入中文显示状态，这就要用到下面一段特别的程序：

```
0  REM  OPEN  THE  C. C. G  I/O
1  HIMEM: 36864: POKE  37984,0: PR# 3:
   PRINT  "  "
2  POKE  43603,3: POKE  43604,195: POKE
   43605,48: POKE  43606,195: POKE 54,189:
   POKE  55,158: POKE  56,129: POKE  57,
   158
3  FOR  I=1  TO  15: GET  A$: NEXT  I
```

将这条程序加在本例程序的开头，就可以完成上述项目的工作。

九、超级汉卡

当读者学会了使用汉字卡（普通汉卡）之后，就会发觉它有不少缺点。例如：所有的浮点BASIC屏幕控制命令均不能使用，取而代之的是一堆十分难记又无章可循的POKE或CALL地址，而这些地址又必须根据汉字卡所插入的具体接口来确定，同时，输入中文字及退出中文输入状态都必须按“CTRL-L”键，而且中文字由二至六码构成（包括空格键），长短不一，不利于文件处理，等等。

所谓超级汉卡，就是为了改进这些缺点而设计的。就外

型及一般使用（不用所配的磁盘）来看，超级汉卡和普通汉卡并无区别。区别在于超级汉卡有所配的磁盘（普通汉卡不能用此磁盘），在该磁盘中，有一条称为“命令解释器”的程序。该程序及十个字符集在内存中占了约7K字节。该磁盘的全部程序约占50K字节。超级汉卡通过命令解释器取得以下改进：

（1）恢复了APPLESOFT中的TAB，HTAB，VTAB，HOME，INVERSE及NORMAL这些主要屏幕控制命令。

（2）增加了英文小写字母功能，有利于文字处理。

（3）英文字母及数字缩小为中文字的1/4，使屏幕可显示$34\times20=680$个字符，比用普通汉字卡时多了一倍。

（4）退出中文输入状态时，可直接按RETURN而不需要按“CTRL-L”。

（5）中文字代码为等长度，便于文件处理，但和普通汉字卡不兼容。

（6）可以用“CTRL-F n”（$0<n<9$）选择十组字符集中的任一组进行显示或打印。

（7）在屏幕下方增加了一行状态显示行，可以显示中文输入字码（自动显示，无须通过CTRL-D控制）、光标的位置（X、Y坐标），所用的字符集和英文大小写状态。

（8）光标会闪烁。

使用超级汉卡时，首先要“引导”它所配的磁盘的内容，调入命令解释器，然后，机器马上进入中文显示状态。这时即可使用超级汉卡。在超级汉卡的中文状态下，按“RESET”键不会退出到TEXT状态。

第十二章　深入一步的程序设计技巧

在前十章的基础上，本章进一步介绍APPLE Ⅱ微型计算机的显示、作图、音乐程序的设计技巧及磁盘技术。每种技巧均配有相应的实际程序，供读者练习和参考。本章的内容有一定的实用性。

第一节　显示的技巧

本节较详细地介绍屏幕显示RAM（内存）与屏幕的相关位置，以便读者直接与眼睛能看到的“内存”（用于显示的RAM）打交道，增强输出显示的功能。

文本显示区，即通常字符（英文字母、数字、特殊符号等）显示所占用的内存区，设在地址1024至2047（$0400至$07FF）区间里。在这区间里的任一个地址单元，均对应于显示屏幕上的一个确定的位置，详见图12-1。

请注意屏幕位置与显示RAM的对应关系，屏幕对应的内存是被分成几段的，每段的地址区间如下：

（列）

1234567890123456789012345678901234567890

每一行的第一个地址								每一行的最后一个地址
1024	（行）1						1	1063
1152	2						2	1191
1280	3						3	1319
1408	4						4	1447
1536	5		1546			1566	5	1575
1664	6						6	1703
1792	7						7	1831
1920	8						8	1959
1064	9						9	1103
1192	10	1197			1212		10	1231
1320	11						11	1359
1448	12						12	1487
1576	13						13	1615
1704	14						14	1743
1832	15					1862	15	1871
1960	16						16	1999
1104	17						17	1143
1232	18						18	1271
1360	19						19	1399
1488	20			1503			20	1527
1616	21						21	1655
1744	22						22	1783
1872	23						23	1911
2000	24						24	2039

看不见的地址：

1144—1151	1656—1663
1272—1279	1784—1791
1400—1407	1912—1919
1528—1535	2040—2047

图 12-1　文本显示区与屏幕位置的关系

可看见{ A段 1024—1063 （或＄0400—＄0427）
B段 1064—1103 （或＄0428—＄044F）
C段 1104—1143 （或＄0450—＄0477）

不可看见：D段 1144—1151 （或＄0478—＄047F）

下面试运行一个程序：

```
10  HOME
20  FOR  A=1704  TO  1743
30  POKE  A,171
40  NEXT  A
```

上程序中的第10句是在文本状态下清屏。第20和40句构成一循环，允许A由1704增加到1743。这些数目代表着APPLE Ⅱ屏幕上一个完整的行（见图12-1）。第30句是依次输入171这个码（ASC Ⅱ码）所代表的符号（即加号"+"）到相应位置上。当执行这个程序时，将会看到一行"+"号出现在屏幕上。这个过程发生得非常快，我们可以加入下面一句来减慢它的执行速度：

```
35  FOR  B=1  TO  20：NEXT  B
```

若要制造一种"+"号移动的幻觉，可以在每一个"+"号输入之后立即将它去除，便可达到这个目的。

例如可打入以下语句：

```
32  POKE  A-1,160
```

160代表一个空格。执行第32行的结果就是用空白来取代前面放入A位置的加号"+"（现在A-1的位置）。

执行上述语句后，可以发现最后一个加号"+"没有被循环去除，这是因为最后一次循环是放进最后一个加号，而将倒数第二个加号去除。如果要去掉最后的加号，可加入以下语句：

```
50   POKE   A-1,160
```

上述的这种显示技巧在许多场合是有用的。例如，某一动点的连续移动，及对角线的移动等。

以上介绍了用POKE语句与显示区直接打交道的方法。实际上，用机器语言程序也可以直接与显示区打交道——直接把数字（ASCⅡ码）安置在所选定的地址单元内。

放入显示区（内存）的数字可以表示许多特定的意义，它可代表APPLE机所用的字符，也可代表APPLE机所了解的指令或其它存贮单元的地址等。

一、利用POKE语句显示计算机符号的程序

下面介绍二个利用POKE语句在屏幕上显示整套的APPLE计算机符号的程序。

第一个程序：

```
NEW
90   HOME
100  FOR  N=0  TO  255
120  POKE  1980,N
125  FOR  J=1  TO  400:NEXT  J
130  NEXT  N
```

执行这个程序，即可将APPLE计算机键盘上所有的符号依序显示在屏幕的中央（即1980的位置上）。

如果要一次显示所有的字符，则需要写一个新的程序（即第二个程序）：

```
NEW
10   HOME
20   N=1024
40   FOR  A=0  TO  255
```

```
60   POKE  N,A
100  N=N+1
120  NEXT  A
```

二、利用PEEK语句查看显示区内容的程序

PEEK也象POKE一样可直接与显示区打交道。但PEEK是"查阅"，而POKE是"存入"。要查看显示区某位置上存放什么符号，可以用PEEK语句把这些信息取出。如下例：

```
100  HOME
120  A=191：REM  QUESTION  MARK
140  N=461：REM  SCREEN  LOCATION
160  GOSUB  1000：REM  TRANSLATION
180  POKE  P,A
200  FOR  N=441  TO  480
220  GOSUB  1000
240  B=PEEK(P)
250  IF  B=160  THEN  280
255  A$=CHR$(B)
260  PRINT  N, B, A$
280  NEXT  N
999  END
1000 M=INT(N/40)+1
1020 R=N-40*(M-1)
1040 R=R-1
1060 IF  M<9  THEN  1100
1080 IF  M<17  THEN  1120
1100 X=56：Y=M-24：GOSUB  2000
```

```
1120 X=0: Y=M-13: GOSUB 2000
1999 REM  P  IS  PROPER  SCREEN
     NUMBER  FOR  POKE
2000  P=984+M*40+X+Y*88+R
2020 RETURN
```

顺便指出，低分辨率图形显示及高分辨率图形显示也与文本（TEXT）显示一样，它们有各自的显示内存区与屏幕上的位置相对应，详见附录七和附录八。

第二节　APPLE Ⅱ高速图形及屏幕绘图的新方法

本节将详细介绍绘制高速活动图形的一种方法，以及在屏幕上编辑高速图形的技巧。

我们知道，在绘制模拟活动图形时，子程序的执行速度是很重要的。这是因为当图形模拟时，图形由一个位置移往另一位置的过程分为四个步骤，它们所需的时间依次是：绘出图形所需的时间；图形停留在屏幕的时间；消除图形所需的时间；图形不在屏上的时间。每当图形移往一个新的位置时，便会将这四个阶段重复一遍，每个阶段所需的时间，决定了图形移动的快慢。如果这个时间掌握得很恰当，图形的转换就会很畅顺，不致出现闪烁的现象。一般地，如果以上四个阶段的总时间不长于人对图形的视觉暂留时间，图形便可顺滑出现于屏幕上，而不致出现闪烁。

要减少图形离开屏幕的时间是不困难的，因为要求图形在另一点出现的操作，是可以在图形仍停留在屏幕时进行的，当图形被消除时，它便可马上出现在新的位置上。绘出和消除图形所需的时间直接由图形子程序所决定，若子程序工作

慢，绘出和消除图形的时间便较长，因此图形在越过屏幕时便出现闪烁了。

一、APPLE Ⅱ机原有的图形表达方法

APPLE Ⅱ机原用的图形子程序工作得很慢，它是由一系列的向量构成的，这些向量会促使某一点发亮，也能指示驱动该点旁边四点中的其中一点。这种方式用来表达简单的单行图形是最好不过了，如图12-2所示。但是，假如要画出如图12-3所示的图形，使用上述的表达方法会出现麻烦，并且效率很低。用 APPLE Ⅱ 机本身的图形子程序绘制如图12-3所示一类图形，通常需要用较多的存贮量。而且当图形面积很大时，向量表的容量就更大了。另外，绘出这些图形所需的时间取决于子程序操作整个图形表的速度。当向量表代表的图形效果不佳，即效率低时，就会浪费很多的绘图

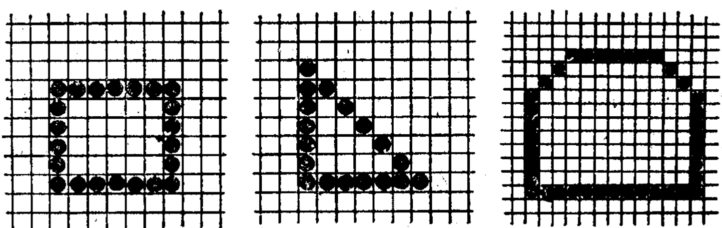


图 12-2 APPLE Ⅱ机本身的图形子程序适合画单线密闭式的图案

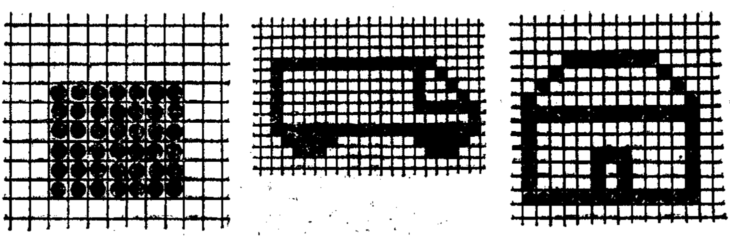


图 12-3 APPLE Ⅱ本身的图形子程序不适宜绘制的图形

时间。

同样，在绘制一些包括有阶梯形状或带有旋转因素的图形时，子程序的速度会降低。因此，为了绘出图形，必须要用更优的子程序来辅助。这些子程序包括比例因素及旋转因素，前者决定绘出图形的大小（实际大小，两倍大小，三倍、四倍等），后者决定图形在绘出前所旋转的角度。这些因素非常实用，但却降低了子程序的操作速度。

二、全新的图形子程序

为了克服机器本身的图形子程序存在的不足之处，建立一个新的图形子程序，应考虑以下两方面的问题。首先，子程序必须能高速操作，以尽量减少图像闪烁的情形；其次，在存贮区内采用表达图形的方法，必须能将复杂的图像快速地显示出来。解决后一个问题的方法是使用"位图素"来表示存贮器内的图形，换句话说，在存贮器中图形的表达方法应和高分辨率的屏幕显示区的表达方法相同。这样，表达图形的字节，只须由主存贮器（RAM）转移到高分辨率显示区，就能快捷简单地绘制出图形来。下面介绍的高速子程序，就是采用这种技术。

由一系列字节组成的表便产生了"位图素"，这种表称为"图形表"（SHAPE TABLE）。读者会注意到，这里的由"位图形点"组成的"图形表"与前面所述的代表"图形向量"的图形表是不同的。我们还将给出一个图形编辑程序，通过它很容易描绘出美观和实用的图形。绘制图形表可以先用一张绘图纸，以一方格代表一个图形点的方式绘出，如图12-4(a)所示。我们以"1"代表亮点，以"0"代表暗点。先绘出一个能包围整个图形但面积最小的矩形，然后将每一行的二进制数字分成7位一组，如图12-4(b)所示。如果最后的一

组数字不足7位，则在每行的左端（高位）或右端（低位）加入0，以凑足7位。由于子程序的限制，每行不能超过7组二进制数字。若将每组二进制数字的位次序倒置，如图12-4(c)所示，就可以用相应的十六进制数字来代替图12-4（b）中的各组二进制数字，得图12-4（d）。除了属于表中顶端的两个字节外，此表大致已经完成了。表顶两个字节中的第一个字节代表图形的高度，即图12-（b）4中的行的数目；第二个字节代表7位组的图形宽度，即在图12-4（b）中每行所使用的7位组的图形的长度。如上所述，图12-4的图形宽度应不大于7组。整个图形表的十六进制数如下：

05 02 78 07 14 04 12 02 11 01 7F 00

我们提供的图形子程序见清单一（见第227页），而清单二（见第235页）是子程序采用的数据表。在运行子程序前，需以一些参数来建立数个寄存器和存贮单元，包括高分辨率屏幕的坐标（它是在左上角位图素位置上的一点）。而较低次序的X坐标的字节应放在X寄存器内，相对应的较高次序的X坐标（不论“1”或“0”）的字节应放在Y寄存器中，而Y坐标放入累加器A中。低次序和高次序的字节在图形表中的开始位置，应贮在十六进地址＄EB和＄EC之中。而子程序可被普通的JSR指令调用。下面绘出来调用图形子程序前要预置的参数值的摘要。

1（a） 坐标	6502寄存器	1（b） 图形表的开始地址	存入地址
X最低字节	X		
X最高字节	Y	地址低位	＄EB
Y	A	地址高位	＄EC

注：“位图素”左上角的坐标（1a），图形表的开始地址（十六进制）（1b）

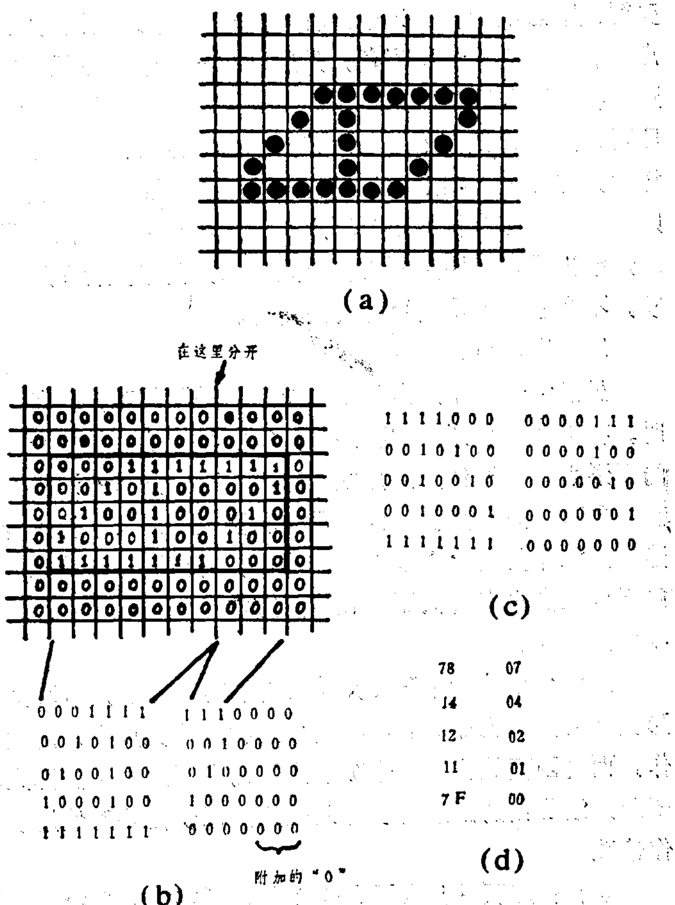


图 12-4 利用绘图纸来作图形。

子程序的工作，是计算在第一页高分辨率显示内存区中受"异或"影响的位。采用这种"异或"技术来绘图有很多优点。首先，无需标明彩色，只须运行一次子程序便可绘出图形；在消除图形时，只需以同一屏幕坐标再调用一遍子程序即可。其次，任何采用"异或"技术所绘制的图形是不会被破坏的，亦即无论图形发生何种变化，在屏幕消失之前它会被贮存起来。这个特性可以利用来创造出很多有趣的背景，在图形绘出后可把图形覆盖在背景之上，而且光标可自由移动，不会破坏屏幕内已存有的内容。

三、图形编辑程序

虽然为一个指定图形编写图形表并不困难，但却十分费时。在编写一个图形的程序时，往往很难确定构成图形所要的图形点数，即使知道点数，亦难以确定图形的形状在高分辨率屏幕内是否美观，因而需要花费较长时间来编出合适的形状。如果先用绘图纸编绘图形，在得出满意的结果时，可能已经耗费很多时间了。较佳的方法是先在屏幕上编绘好所要求的图形。

后面给出的清单三（见第287页），就是利用屏幕实现图形编辑功能的程序。它能在高分辨状态下完成图形的编辑工作，而且画面能出现一个被放大了的图形及一个实际大小的图形，绘画出的图形可以存入磁盘，以便需要时使用或取出作修改。

编辑程序要使用APPLE Ⅱ机32K字节的存贮量，同时要配备一台磁盘机和FP-BASIC语言。当运行程序时，屏幕上会显示出如下信息：

COMMAND MENU

I, J, K, M	CURSOR MOVEMENT
P	PLOT POINT AT CURSOR POSITION
E	ERASE POINT AT CURSOR POSITION
C	CLEAR SCREEN
T	MAKE SHAPE TABLE
S	SAVE SHAPE SOURCE FILE TO DISK
G	GET SHAPE SOURCE FILE FROM DISK
H CR?	HELP (RETURN TO THIS MENU)
Q	QUIT PROGRAM EXECUTION
	HIT SPACE TO EXIT MENU

任何时候只要按下H键，就会返回这个索引（菜单）。

按下SPACE键（空格）后，屏幕左边位置会转黑，而右边则出现一个方框。漆黑的左部是用来绘制实际大小的图形的，方框中则是图形放大了的情形。当方框出现后，其中一方格会出现一细小的水平线，这就是光标，用来表示程序提供绘图的位置。

当光标出现后，便可执行屏幕上给定的命令（只须按下所需要的字母便可实现）：

I，J，K，M可分别使光标向上、左、右、下移动，

即

```
        I
        ↑
    J ←——→ K
        ↓
        M
```

P（LOT）：绘出光标位置一点（亮）；

E（RASE）：消除光标位置上一点（暗）；

C（LEAR）：清除整个屏幕的图形。

利用上述命令，即可以画出所需的图形。画完后，把光标移至图形的左上角（不是屏幕的左上角），见图12-6。图12-5为编绘中的屏幕。

T键：做出一个图形表，即选出一个最小的矩形来包围图形（正如上述用人工建立形状表时所建立的矩形）。实际上可以通过移动光标来获得围绕这个图形的矩形。操作方法如下：把光标移到右下角，按下RETURN键，使光标移到左上角时按下RETURN键，便可确定该矩形。在选择好所需的矩形后，程序便会作出图形表。所需的时间约15～30秒之间，视图形的大小而异。完成后的图形表，会以十六进或十进的形式显示在屏幕上。图12-7就是显示出最后的图形表。

图形表是一般的二进制文件。通过键操作，可把它的程序当作目标文件（OBJECT-FILE）存入磁盘中。

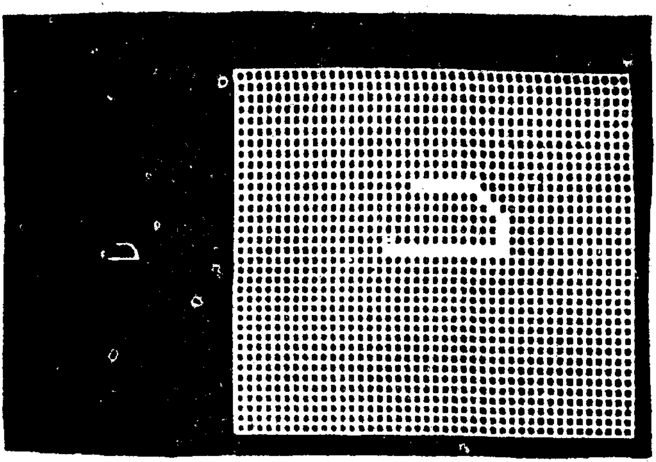
S（AVE）：将已编绘好的图形表（OBJECT-FILE）存入磁盘，并可选定文件名。

G（ET）：将磁盘中的图形表放入内存，供使用或再修改。

H(ELP)，或？键：可回到主目录（菜单），供选择命令。

空格键：回到屏幕编绘的状态，而原来的图像不改变。

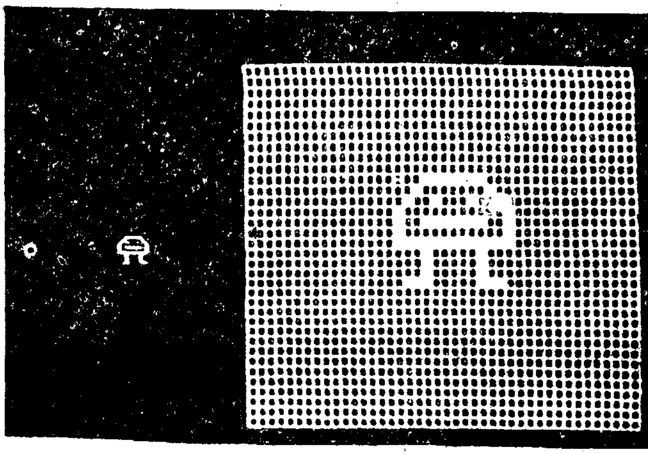
Q(UIT)：结束(退出)该图形编辑程序，同时使屏幕上的图形消失。所以编绘过程中不应按Q键。



实际尺寸　　　　放大了的图形

图 12-5　图形编绘程序的一屏幕

左边的图形是实际大小，右边是放大了的图形



实际尺寸　　　　　　　　　　放大了的图形

图 12-6　绘成图形后，将光标移至左上角

```
DO YOU WANT TO SEE THE TABLE
   IN HEX OR IN DECIMAL?
   0B 02 78 07 04 08 12 12
   01 20 79 27 01 20 7F 3F
   08 04 08 04 08 04 0E 1C
DO YOU WANT TO SAVE THE OBJECT
   TABLE ON DISK?
WHAT DO YOU WANT TO NAME
   THE FILE? █
```

图 12-7　完成绘图后跟着出现图形表

采用这种新方法——“位图素”图形表及屏幕编辑法，可以使读者得到专业性实用性都较好的图形效果，它并不受原 APPLE Ⅱ 机图形子程序(慢速)所限制。把这种新方法灵活运用在高分辨率屏幕上，就可以绘出各种美观的图形。

清单一：用于绘制APPLE Ⅱ高分辨图形的快速图形子程序

```
0000:        OBJ   $1B00
1B00:        ORG $1B00   ; ASSEMBLY
                           LOCATION
1B00:   ********************
1B00:   *                  *
1B00:   * SHAPE WORKS BY STEPPING *
1B00:   * THROUGH THE USER TABLE *
1B00:   * ONE HI-RES LINE AT A   *
1B00:   * TIME, SHIFTING THE BIT *
1B00:   * PATTERN THE APPROPRIATE *
1B00:   * NUMBER OF TIMES (DEPEND- *
1B00:   * ING ON THE X-COORDINATE *
1B00:   * PASSED IN THE X- AND Y- *
1B00:   * REGISTERS), AND MOVING *
1B00:   * THE PATTERN TO THE PRO- *
1B00:   * PER PLACE IN THE HI-RES *
1B00:   * SCREEN MEMORY.         *
1B00:   ********************
1B00:   STARTZ EQU  $19    ; START
                              OF LINE
                              STORAGE
1B00:   YCOORD  EQU  $E3  ; LINE
                              COUNTER
1B00:   START  EQU$  EB ; USER
                         TABLE POINTER
1B00:   ADDRL  EQU  $ED  ; 1ST
                           SCREEN BYTE
                              TO USE
```

```
1B00 :          ADDRH   EQU   $EE  ; IN LINE
                                     YCOORD
1B00 :          ADDRADD EQU   $EF  ; OFF-
                                     SET FORM
                                     LEFT BYTE
1B00 :          SHFTNUM  EQU  $F9  ; NUM-
                                     BER OF
                                     SHIFTS
1B00 :          ENDLN   EQU   $FD  ; LAST
                                     LINE + 1
1B00 :          WIDTH   EQU   $FB  ; WIDTH
                                     OF USER
                                     TABLE
1B00 :          INDEX   EQU   $FC  ;POINTER
                                     IN USER
                                     TABLE
1B00 :          *
1B00 :          *  DIVIDE X-COORD BY 7 TO
1B00 :          *  GET BYTE OFFSET FROM
1B00 :          *  LEFTMOST BYTE IN ANY HI-
1B00 :          *  RES LINE.REMAINDER WILL
1B00 :          *  BE CORRECT NUMBER OF
1B00 :          *  SHIFTS TO PERFORM ON BIT
1B00 :          *  PATTERN. DIVISION IS PER-
1B00 :          *  FORMED USING LOOKUP
                   TABLE
1B00 :          *  FOR SPEED.
1B00 :          *
1B00 : 85 E3          STA YCOORD ; STORE Y-
```

```
                                          COORD
                                          (COUNTER)
1B02 : 8A          TXA
1B03 : 0A          ASL  A
1B04 : AA          TAX
1B05 : 98          TYA
1B06 : 2A          ROL  A
1B07 : A8          TAY              ; MULTIPLY
                                          X-COORD
                                          BY  TWO
1B08 : 18          CLC
1B09 : 8A          TXA              ; A-REG = X-
                                    COORD   * 2
                                    LO-BYTE
1B0A : 69 83    ADC   #>QUOTBL;ADD TABLE
                                    ADDRESS LO-
                                          BYTE
1B0C : 85 ED       STA ADDRL     ; STORE
                                          RESULT
1B0E : 98          TYA    ;       A-REG = X-
                                          COORD * 2
                                          HI-BYTE
1B0F : 69 1B      ADC   # <QUOTBL;ADD TABLE
                                          ADDRESS
                                          HIBYTE
1B11 : 85 EE       STA  ADDRH  ; STORE
                                          RESULT
1B13 : A0 00      LDY   # $00          ZERO Y-
                                          REG FOR
```

```
                                        INDEXING
1B15 : B1   ED     54      LDA(ADDRL), Y   ; LOAD
                                        X-COORD/7
                                        FROM TABLE
1B17 : 85 EF       55      STA   ADDRADD;ADDRADD
                                        =X-COORD/7
1B19 : C8          56        INY          ; REMAINDER
                                        FOLLOWS
                                        IN  TABLE
1B1A : B1   ED     57      LDA  (ADDRL),  Y  ; LOAD
                                        REMAINDER
                                        FROM TABLE
1B1C : 85 F9       58       STA SHFTNUM; SHFTNUM
                                        =REMAIN
                                        DER
1B1E :             59  *
1B1E :             60  * INITIALIZE LOCATIONS END-
1B1E :             61  *  LN AND WIDTH,ENDLN CON-
1B1E :             62  *  TAINS THE Y COORD OF THE
1B1E :             63  *  LAST LINE+1,  WIDTH CON-
1B1E :             64  * TAINS THE WIDTH(IN BYTES)
1B1E :             65  * OF EACH LINE.
1B1E :             66  *
1B1E : A5 E3       67       LDA YCOORD
1B20 : A0 00       68      LDY   #$00
1B22 : 18          69      CLC
1B23 : 71 EB       70      ADC (START), Y
1B25 : 85 FD       71      STA ENDLN     , ENDLN=Y-
                                        COORD+
```

```
                                         LENGTH
1B27 : C8                INY
1B28 : B1 EB             LDA (START), Y
1B2A : 85 FB             STA WIDTH   ; GET&STORE
                                        WIDTH
1B2C : C8                INY
1B2D : 84 FC             STY INDEX   ; INDEX=2
1B2F :                *
1B2F :                *  LOOP1 IS THE LOOP THAT IS
1B2F :                *  CYCLED THROUGH ONCE FOR
1B2F :                *  EACH LINE ON THE HI-RES
1B2F :                *  SCREEN
1B2F :                *
1B2F : A6 FB          LOOP1 LDX WIDTH,    X-REG=0
                                        (COUNTER)
1B31 : A4   FC              LDY INDEX
1B33 :                *
1B33 :                *  MOVE BYTES FOR LINE
1B33 :                *  YCOORD FROM USER TABLE
1B33 :                *  TO ZERO PAGE
1B33 :                *
1B33 : B1   EB        LOOP2 LDA (START),Y ; GET
                                          XTH
                                          BYTE
                                        OF LINE
1B35 : 95   19              STA STARTZ,X; STORE
                                      IN STARTZ
                                        +X
1B37 : C8                   INY
```

```
1B38 : CA                   DEX           MOVED ALL
                                          BYTES YET?
1B39 : D0   F8              BNE LOOP2   ;NO, LOOP
1B3B : 86   19              STX STARTZ  ;STARTZ = 0
1B3D : 84   FC              STY INDEX
1B3F :              *
1B3F :              *  SHIFT THE BIT PATTERN
1B3F :              *  SHFTNUM TIMES
1B3F :              *
1B3F : A4   F9              LDY SHFTNUM ;IS SHFT-
                                          NUM = 0?
1B41 : F0   16              BEQ SKIP    ;YES, SKIP
                                          THE SHIFT-
                                          ING
1B43 : 18           LOOP3   CLC         ;NO, START
                                          SHIFTING
1B44 : A6   FB              LDX WIDTH
1B46 : 08                   PHP         ;KEEP STACK
                                          IN ORDER
1B47 : 28           LOOP4   PLP         ;RESTORE
                                          CARRY
1B48 : B5   19              LDA STARTZ,X ;LOAD
                                          ORIGINAL
                                          PATTERN
1B4A : 2A                   ROL A
1B4B : 2A                   ROL A       ;ROTATE
                                          LEFT TWICE
1B4C : 08                   PHP         ;SAVE CARRY
1B4D : 4A                   LSR A       ;SHIFT RIGHT
```

```
                                  ONCE
1B4E : 95    19        STA STARTZ,X   ;STORE
                                       SHIFT-
                                      ED PAT-
                                        TERN
1B50 : CA              DEX
1B51 : E0 FF           CPX #$FF     ;ROTATED
                                   EACH BYTE?
1B53 : D0    F2        BNE LOOP4     ;NO,LOOP
1B55 : 28              PLP           ;KEEP
                                     STACK IN
                                     ORDER
1B56 : 88              DEY
1B57 : D0   EA         BNE LOOP3      ;LOOP IF
                                      Y<  >0
1B58 :            *
1B58 :            *  CALCULATE HI-RES
                                     SCREEN
1B58 :            *  ADDRESS FOR FIRST BYTE
1B59 :            *  TO BE USED IN LINE
1B59 :            *  YCOORD
1B59 :            *
1B59 : A4   E3     SKIP LDY YCOORD
1B60 : B9  B3 1D        LDA LOSTRT,Y
1B5E : 18               CLC
1B5F : 65   EF          ADC ADDRADD
1B61 : 85   ED          STA ADDRL
1B63 : B9  73 1E        LDA HISTRT,Y
1B66 : 69   00          ADC  #$00
```

```
1B68 : 85   EE     132          STA ADDRH ;GET ADDR
                                              FOR 1ST
                                              BYTE
1B6A :             133     *
1B6A :             134     *   MOVE SHIFTED BYTES
                                              FROM
1B6A :             135     *   ZERO PAGE TO HI-RES
1B6A :             136     *   SCREEN MEMORY. FOR NON-
1B6A :             137     *   EXCLUSIVE-OR PLOTTING,
1B6A :             139     *   CHANGE EOR(ADDRL),Y'
1B6A :             140     *   TO 'ORA ( ADDRL ), Y'
1B6A :             141     *   ( OPCODE  $11 ).
1B6A :             142     *
1B6A : A0   00     143          LDY  # $00
1B6C : A6   FB     144          LDX WIDTH
1B6E : B5   19     146    LOOP5 LDA STARTZ,X
1B70 : 51   ED     147          EOR ( ADDRL ),Y
1B72 : 91   ED     148          STA ( ADDRL ),Y  ;PLOT
                                                  BYTE
                                                  ON
                                                  SCREEN
1B74 : C8          149          INY
1B75 : CA          150          DEX
1B76 : E0   FF     151          CPX  # $FF  ,THROUGH
                                            PLOTTING
                                            LINE?
1B78 : D0   F4     152          BNE LOOP5    ;NO, LOOP
1B7A : E6   E3     153          INC YCOORD    ;YES, GO
                                              TO NEXT
```

```
                                            LINE
1B7C:A5   E3          LDA  YCOORD
1B7E:C5   FD          CMP  ENDLN     ;MORE
                                      LINES?
1B80:D0   AD          BNE  LOOP1    ;YES,LOOP
1B82:60               RTS           ;NO,RE-
                                      TURN
1B83:           QUOTBL  EQU  *
1B83:           LOSTRT  EQU  * +560
1B83:           HISTRT  EQU  * +752
```

清单二：高级绘图子程序所需用到的数据表

```
1B83-  00 00 00 01 00 02 00 03 00 04 00 05 00
1B90-  06 01 00 01 01 01 02 01 03 01 04 01 05 01 06 02
1BA0-  00 02 01 02 02 02 03 02 04 02 05 02 06 03 00 03
1BB0-  01 03 02 03 03 03 04 03 05 03 06 04 00 04 01 04
1BC0-  02 04 03 04 04 04 05 04 06 05 00 05 01 05 02 05
1BD0-  03 05 04 05 05 05 06 06 00 06 01 06 02 06 03 06
1BE0-  04 06 05 06 06 07 00 07 01 07 02 07 03 07 04 07
1BF0-  05 07 06 08 00 08 01 08 02 08 03 08 04 08 05 08
1C00-  06 09 00 09 01 09 02 09 03 09 04 09 05 09 06 0A
1C10-  00 0A 01 0A 02 0A 03 0A 04 0A 05 0A 06 0B 00 0B
1C20-  01 0B 02 0B 03 0B 04 0B 05 0B 06 0C 00 0C 01 0C
1C30-  02 0C 03 0C 04 0C 05 0C 06 0D 00 0D 01 0D 02 0D
1C40-  03 0D 04 0D 05 0D 06 0E 00 0E 01 0E 02 0E 03 0E
1C50-  04 0E 05 0E 06 0F 00 0F 01 0F 02 0F 03 0F 04 0F
1C60-  05 0F 06 10 00 10 01 10 02 10 03 10 04 10 05 10
1C70-  06 11 00 11 01 11 02 11 03 11 04 11 05 11 06 12
1C80-  00 12 01 12 02 12 03 12 04 12 05 12 06 13 00 13
1C90-  01 13 02 13 03 13 04 13 05 13 06 14 00 14 01 14
```

```
1CA0-  02 14 03 14 04 14 05 14 06 15 00 15 01 15 02 15
1CB0-  03 15 04 15 05 15 06 16 00 16 01 16 02 16 03 16
1CC0-  04 16 05 16 06 17 00 17 01 17 02 17 03 17 04 17
1CD0-  05 17 06 18 00 18 01 18 02 18 03 18 04 18 05 18
1CE0-  06 19 00 19 01 19 02 19 03 19 04 19 05 19 06 1A
1CF0-  00 1A 01 1A 02 1A 03 1A 04 1A 05 1A 06 1B 00 1B
1D00-  01 1B 02 1B 03 1B 04 1B 05 1B 06 1C 00 1C 01 1C
1D10-  02 1C 03 1C 04 1C 05 1C 06 1D 00 1D 01 1D 02 1D
1D20-  03 1D 04 1D 05 1D 06 1E 00 1E 01 1E 02 1E 03 1E
1D30-  04 1E 05 1E 06 1F 00 1F 01 1F 02 1F 03 1F 04 1F
1D40-  05 1F 06 20 00 20 01 20 02 20 03 20 04 20 05 20
1D50-  06 21 00 21 01 21 02 21 03 21 04 21 05 21 06 22
1D60-  00 22 01 22 02 22 03 22 04 22 05 22 06 23 00 23
1D70-  01 23 02 23 03 23 04 23 05 23 06 24 00 24 01 24
1D80-  02 24 03 24 04 24 05 24 06 25 00 25 01 25 02 25
1D90-  03 25 04 25 05 25 06 26 00 26 01 26 02 26 03 26
1DA0-  04 26 05 26 06 27 00 27 01 27 02 27 03 27 04 27
1DB0-  05 27 06 00 00 00 00 00 00 00 00 80 80 80 80 80
1DC0-  80 80 80 00 00 00 00 00 00 00 00 80 80 80 80 80
1DD0-  80 80 80 00 00 00 00 00 00 00 00 80 80 80 80 80
1DE0-  80 80 80 00 00 00 00 00 00 00 00 80 80 80 80 80
1DF0-  80 80 80 28 28 28 28 28 28 28 28 A8 A8 A8 A8 A8
1E00-  A8 A8 A8 28 28 28 28 28 28 28 28 A8 A8 A8 A8 A8
1E10-  A8 A8 A8 28 28 28 28 28 28 28 28 A8 A8 A8 A8 A8
1E20-  A8 A8 A8 28 28 28 28 28 28 28 28 A8 A8 A8 A8 A8
1E30-  A8 A8 A8 50 50 50 50 50 50 50 50 D0 D0 D0 D0 D0
1E40-  D0 D0 D0 50 50 50 50 50 50 50 50 D0 D0 D0 D0 D0
1E50-  D0 D0 D0 50 50 50 50 50 50 50 50 D0 D0 D0 D0 D0
1E60-  D0 D0 D0 50 50 50 50 50 50 50 50 D0 D0 D0 D0 D0
```

```
1E70-  D0 D0 D0 20 24 28 2C 30 34 38 3C 20 24 28 2C 30
1E80-  34 38 3C 21 25 29 2D 31 35 39 3D 21 25 29 2D 31
1E90-  35 39 3D 22 26 2A 2E 32 36 3A 3E 22 26 2A 2E 32
1EA0-  36 3A 3E 23 27 2B 2F 33 37 3B 3F 23 27 2B 2F 33
1EB0-  37 3B 3F 20 24 28 2C 30 34 38 3C 20 24 28 2C 30
1EC0-  34 38 3C 21 25 29 2D 31 35 39 3D 21 25 29 2D 31
1ED0-  35 39 3D 22 26 2A 2E 32 36 3A 3E 22 26 2A 2E 32
1EE0-  36 3A 3E 23 27 2B 2F 33 37 3B 3F 23 27 2B 2F 33
1EF0-  37 3B 3F 20 24 28 2C 30 34 38 3C 20 24 28 2C 30
1F00-  34 38 3C 21 25 29 2D 31 35 39 3D 21 25 29 2D 31
1F10-  35 39 3D 22 26 2A 2E 32 36 3A 3E 22 26 2A 2E 32
1F20-  36 3A 3E 23 27 2B 2F 33 37 3B 3F 23 27 2B 2F 33
1F30-  37 3B 3F
```

清单三：图形编辑程序

```
100 TEXT : HOME : POKE -16298, 0 : POKE -16300, 0
110 RESTORE : FOR I=768 TO 774 : READ J : POKE
    I, J : NEXT I : POKE 232,0 : POKE 233, 3 : DATA
    1, 0, 3, 0, 45, 5, 0
120 DIM S%(105), T%(212)
130 XMAX=42 : YMAX=35 : ML=101 : MT=10
140 H$ ="0123456789ABCDEF"
150 D$ =CHR$ (4)
160 GOSUB 3100 : GOSUB 3300 : GOSUB 3400
400 REM SHOW CURSOR POSITION ON GRID
410 XDRAW 1 AT CL+1,CT+ 3
420 REM WAIT FOR KEYBOARD COMMAND
430 Q=PEEK( -16384) : IF Q < 128 THEN 430
440 POKE -16368, 0 : Q=Q-128
```

```
500  REM
501  REM  CURSOR MOVEMENT COMMANDS
502  REM
510  IF Q<  > ASC("I")THEN 550
520  XDRAW 1 AT CL+1,CT+ 3
530  IF Y > 1 THEN Y=Y- 1: CT=CT- 4
540  GOTO 410
550  IF Q <  > ASC ("M") THEN 590
560  XDRAW 1 AT CL+1,CT+ 3
570  IF Y < YMAX THEN Y=Y+ 1: CT=CT+ 4
580  GOTO 410
590  IF Q <  > ASC("J") THEN 630
600  XDRAW 1 AT CL+1,CT+ 3
610  IF X > 1 THEN X=X- 1 : CL=CL- 4
620  GOTO 410
630  IF Q <  > ASC ("K") THEN 700
640  XDRAW 1 AT CL+1,CT+ 3
650  IFX < XMAX THEN X =X+ 1: CL=CL+ 4
660  GOTO 410
700  REM
701  REM  PLOT COMMAND
702  REM
710  IF Q <  > ASC("P") THEN 810
720  ELE=INT ((X- 1 )/14) + 3 * (Y- 1 )
730  BIT= (X- 1 ) -INT((X- 1 )/14) *14
740  A=INT (S%(ELE) / 2 ^ BIT)
750  IF A/2 <  > INT (A/2 ) THEN 810
760  S%(ELE) =S%(ELE) + 2 ^BIT
770  FOR I= 2 TO 4: XDRAW 1 AT CL+1, CT+I:
```

```
    NEXT I
780 HCOLOR= 7 : HPLOT 29+X, 62+Y
790 GOTO 430
800 REM
801 REM ERASE COMMAND
802 REM
810 IF Q < > ASC("E") THEN 900
820 ELE=INT ((X-1)/14)+3*(Y-1)
830 BIT=(X-1)-INT ((X-1)/14)*14
840 A=INT (S%(ELE)/2^BIT)
850 IF A/2 =INT(A/2) THEN 900
860 S%(ELE)=S%(ELE)-2^BIT
870 FOR I = 2 TO 4: XDRAW 1 AT CL+1,CT+I:
    NEXT I
880 HCOLOR= 0 : HPLOT 29+X, 62+Y
890 GOTO 430
900 REM
901 REM CLEAR SCREEN COMMAND
902 REM
910 IF Q < > ASC("C") THEN 1030
920 XDRAW 1 AT CL+1, CT+3
930 VTAB 23: PRINT "SURE YOU WANT TO ERASE
    THE SCREEN?"
940 GOSUB 3500
950 VTAB 22: CALL -958: IF Q < > ASC("Y") THEN
    410
960 FOR I = 0 TO 105: S%(I) = 0 : NEXT I
970 GOSUB 3300 : GOSUB 3400 : GOTO 410
1000 REM
```

```
1010  REM TABLE COMMAND
1020  REM
1030  IF Q < > ASC ("T") THEN 1520
1040  VTAB 23: PRINT "SET CURSOR TO TOP LEFT
      CORNER OF": PRINT"DESIRED BIT MAP AND
      HIT RETURN";
1050  L5 = 1
1060  GOTO 430
1070  PL = X : PT = Y
1080  VTAB 22 : CALL - 958 : PRINT : PRINT"SET CUR
      SOR TO BOTTOM RIGHT CORNER OF": PRINT
      "DESIRED BIT MAP AND HIT RETURN" ;
1090  L5 = 2
1100  GOTO 430
1110  PR = X : PB = Y : L5 = 0
1120  XDRAW 1 AT CL + 1, CT + 3
1130  VTAB 22 : CALL - 958
1140  IF PL > PR OR PT > PB THEN VTAB 23 :
      HTAB 1 : POKE 50, 63 : PRINT "ILLEGAL BIT
      MAP CORNERS" : POKE 50, 255 : FOR I = 1 TO
      2000: NEXT I : VTAB 22 : CALL - 958 : GOTO 410
1150  VTAB 23 : HTAB 1 : PRINT "NOW FORMING
      SHAPE TABLE"
1160  FOR I = 0 TO 212 : T%(I) = 0 : NEXT I
1170  L = PB - PT + 1 : W = (PR - PL + 1) / 7 : IF W < >
      INT (W) THEN W = INT (W) + 1
1180  T%(0) = L : T%(1) = W : N = 2 : Q = 0
1190  FOR Y = PT TO PB
1200  FOR X = PL TO PL + W * 7 - 1
```

```
1210  IF X > PR THEN BN=0 : GOTO 1250
1220  ELE=INT ((X-1) / 14)+3*(Y-1)
1230  BIT=(X-1)-INT ((X-1) / 14)*14
1240  BN=0 : A=INT (S%(ELE) / 2 ^ BIT) : IF INT
      (A/2) <  > A/2 THEN BN=1
1250  IF BN=1 THEN T%(N)=T%(N)+2^Q
1260  Q=Q+1 : IF Q=7 THEN Q=0 : N=N+1
1270  NEXT X : NEXT Y
1280  HOME : POKE-16303,0
1290  VTAB 2 : PRINT "DO YOU WANT TO SEE
      THE TABLE IN HEX" : P RINT" OR IN
      DECIMAL?" : PRINT : PRINT
1300  GOSUB 3500
1310  IF Q<  > ASC ("D") AND  Q <  > ASC ("H")
      THEN 1280
1320  Z=0 : FOR I=0 TO L*W+1
1330  Z=Z+1
1340  IF Q=ASC ("D") THEN PRINT TAB(Z*4);T%
      (I); : GOTO 1360
1350  PRINT TAB(Z*3) ; MID$ (H$,INT (T%(I) /
      16)+1,1) ; MID$ (H$,T%(I)-INT (T%(I) / 16)
      *16+1,1);
1360  IF Z=8 THEN Z =0 : PRINT
1370  NEXT I
1380  PRINT : PRINT : IF PEEK (37) <21 THEN POKE
      34,PEEK (37)
1390  PRINT "DO YOU WANT TO SAVE THE OBJE-
      CT TABLE" : PRINT "ON DISK?"
1400  GOSUB 3500
```

```
1410  IF Q <  > ASC ("Y") THEN 1470
1420  PRINT : PRINT "WHAT DO YOU WANT TO
      NAME" : INPUT " THE FILE?" ; N$
1430  FOR I=0 TO L * W+1 : POKE 16384+I, T%(I) :
      NEXT I
1440  PRINT D$ ; "BSAVE" ; N$ ; ", A$4000, L";
      L * W+2
1450  PRINT "FILE SAVED USING NAME " ; N$
1460  PRINT : PRINT : GOTO 1390
1470  POKE 34,0 : HOME : VTAB 2 : PRINT "DO YOU
      WANT TO RETURN TO THE" : PRINT"
      SCREEN EDIT MODE?"
1480  GOSUB 3500
1490  IF Q <  > ASC ("Y") THEN 2260
1500  GOSUB 3100 : POKE -16304,0 : GOSUB 3310 : GO
      TO 410
1510  REM' RETURN' PSEUDO-COMMAND
1520  IF Q <  > 13 THEN 1600
1530  ON L5+1 GOTO 430, 1070, 1110
1600  REM
1601  REM SAVE TABLE COMMAND
1602  REM
1610  IF Q<  > ASC ("S") THEN 1800
1620  XDRAW 1 AT CL+1,CT+3
1630  VTAB 23 : INPUT "FILE NAME FOR SAVE?";
      N$
1640  VTAB 24 : PRINT "NOW SCANNING IMAGE";:
      HTAB 1
1650  Z1=0
```

```
1660 IF S%(Z1) = 0 AND Z1 < 105 THEN Z1 = Z1 + 1 :
     GOTO 1660
1670 Z2 = 105
1680 IF S%(Z2) = 0 AND Z2 > 0 THEN Z2 = Z2 - 1 : GO-
     TO 1680
1690 IF Z1 > Z2 THEN Z1 = 0 : Z2 = 1
1700 VTAB 24 : PRINT "NOW SAVING IMAGE TO
     DISK";: VTAB 23 : PRINT
1710 PRINT D$; "OPEN"; N$ : PRINT D$;
     "WRITE"; N$
1720 PRINT Z1 : PRINT Z2
1730 FOR I = Z1 TO Z2
1740 PRINT S%(I)
1750 NEXT I
1760 PRINT D$; "CLOSE", N$
1770 VTAB 22: CALL-958 : GOTO 410
1800 REM
1801 REM LOAD TABLE COMMAND
1802 REM
1810 IF Q < > ASC ("G") THEN 2100
1820 XDRAW 1 AT CL + 1, CT + 3
1830 VTAB 23: PRINT "SURE YOU WANT TO
     LOAD?"
1840 GOSUB 3500
1850 VTAB 22: CALL - 958 : IF Q < > ASC ("Y")
     THEN 410
1860 VTAB 23: INPUT "FILE NAME FOR LOAD?",N$
1870 PRINT D$,"OPEN",N$: PRINT D$,"READ",
     N$
```

```
1880 INPUT Z1: INPUT Z2
1890 FOR I = 0 TO Z1:S%(I) = 0: NEXT I: FOR
     I = Z2 TO 105: S%(I) = 0: NEXT I
1900 FOR I = Z1 TO Z2
1910 INPUT S%(I)
1920 NEXT I
1930 PRINT D$;"CLOSE";N$
1940 GOSUB 3300: GOSUB 3400
1950 VTAB 22: CALL - 958: VTAB 23: PRINT
     "NOW RETRACING IMAGE ON SCREEN"
1960 ELE = Z1: BIT = 0:CL =ML+4*((ELE-INT
     (ELE / 3)*3)*14)
1970 CT=MT+4*INT (ELE / 3)
1980 A=INT (S%(ELE) / 2 , BIT):
1981 IF INT (A / 2) = A / 2 THEN 2000
1990 FOR I = 2 TO 4: XDRAW 1 AT CL+1,CT+1:
     NEXT I:HPLOT 30+(CT-ML) / 4,63+(CT-
     MT) / 4
2000 CL=CL+4:BIT=BIT+1:IF BIT < > 14 THEN
     1980
2010 IF ELE > = Z2 THEN GOSUB 3310: GOTO 410
2020 BIT=0:ELE=ELE+1
2030 IF ELE / 3=INT (ELE / 3) THEN CL=ML:CT
     =CT+4
2040 GOTO 1980
2100 REM
2101 REM HELP COMMAND
2102 REM
2110 IF Q < > ASC ("H") AND Q < >ASC ("/")
```

```
     AND Q <  > ASC ("?") THEN 2200
2120 VTAB 21: CALL-958: POKE-16303,0
2130 GOSUB 3170
2140 POKE-16304,0
2150 VTAB 20: PRINT : CALL-958 : HTAB 2: PRINT
     "ACTUAL SIZE";: HTAB 21: PRINT "VIEWING
     WINDOW"
2160 GOTO 430
2200 REM
2201 REM QUIT COMMAND
2202 REM
2210 IF Q <  > ASC ("Q") THEN 430
2220 XDRAW 1 AT CL+1,CT+3
2230 VTAB 23: PRINT "SURE YOU WANT TO
     QUIT?"
2240 GOSUB 3500
2250 IF Q <  > ASC("Y")THEN VTAB 22: CALL-
     958: GOTO 410
2260 HOME : POKE-16303,0: POKE-16298,0: VTAB
     24
2270 GOTO 9999
3000 REM
3010 REM SUBROUTINES
3020 REM
3100 HOME
3110 HTAB 15: PRINT "COMMAND MENU": HTAB
     15: PRINT "------------"
3120 VTAB 4: PRINT "I,J,K,M"; TAB( 9);"CURSOR
     MOVEMENT": PRINT : PRINT "P"; TAB( 9);
```

```
"PLOT POINT AT CURSOR POSITION": PRINT
3130 PRINT "E"; TAB( 9);"ERASE POINT AT
     CURSOR POSITION": PRINT : PRINT "C";
     TAB( 9);"CLEAR SCREEN": PRINT
3140 PRINT "T"; TAB( 9);"MAKE SHAPE
     TABLE": PRINT: PRINT "S"; TAB ( 9);
     "SAVE SHAPE SOURCE FILE TO DISK":
     PRINT
3150 PRINT "G"; TAB ( 9);"GET SHAPE SOURCE
     FILE FROM DISK": PRINT "H OR?"; TAB
     ( 9); "HELP(RETURN TO THIS MENU)"
3160 PRINT : PRINT "Q"; TAB( 9); "QUIT
     PROGRAM EXECUTION"
3170 VTAB 24: HTAB 10: PRINT "HIT SPACE TO
     EXIT MENU";
3180 GOSUB 3500: IF Q < > ASC (" ") THEN
     3180
3190 VTAB 21: CALL -958
3200 RETURN
3300 POKE 230,32: CALL 62450: HGR : SCALE=1:
     ROT=0
3310 PT=YMAX+1:PB=0:PL=XMAX+1:PR=0
3320 VTAB 21: HTAB 2: PRINT "ACTUAL SIZE";:
     HTAB 21: PRINT "VIEWING WINDOW";: CALL
     -958: PRINT
3330 X=INT (XMAX / 2):Y=INT (YMAX / 2)
3340 MR=ML+XMAX * 4:MB=MT+YMAX * 4
3350 CL=ML+(X-1) * 4:CT=MT+(Y-1) * 4
3360 RETURN
```

```
3400 HCOLOR=7
3410 FOR I=ML TO MR STEP 4: HPLOT I, MT TO
     I, MB:NEXT I
3420 FOR I=MT TO MB STEP 4: HPOLT ML,I TO
     MR, I:NEXT I
3430 RETURN
3500 Q=PEEK (-16384): IF Q < 128 THEN 3500
3510 POKE-16368,0:Q=Q-128
3520 RETURN
9999 END
```

第三节　编制音乐程序的技巧

本节将介绍CALL语句与音乐的关系。

一、发出一个单音（可调整音调、节拍）的程序

```
10 HOME
100 DATA 172, 1, 8, 174, 1, 8, 169,4,32,168,
    252, 173, 48, 192, 232, 208, 253, 186, 208,
    239, 206, 0, 8
110 DATA 208, 231, 96, 32, 32, 112, 8, 24,
    216, 136, 8, 160, 160, 16, 56
120 FOR N=770 TO 807
140 READ Y
160 POKE N, Y
180 NEXT N
200 PRINT:INPUT "WHAT PITCH(0—255)";
    F
220 PRINT:INPUT "WHAT DURATION(0—
```

```
      255)"; D
240  POKE  768, D
260  POKE  769, F
300  CALL  770
```

以上程序中的第100和110两行是设置一个用机器码编写的发音子程序，该程序可以被CALL语句调用，执行后可返回。第120～180行则把以上23个参数存入地址770～792中。第200行键入调整“音节”数P；第220行键入“节拍”（时间）数D；第240，260行分别把P、D存入两个特定的地址。它们的作用如下：

地址768　　存放单音的音调

地址769　　存放单音的时间或长度

第300句是调用存放于地址（入口）770的机器语言发音程序。因而执行上述程序时，会听到发出一个单音，而且音调及发音长短（节拍）将由输入的F值和D值来决定。

以上是发出一个单音的程序。如果预先按照乐曲的音符及节拍编置好一些数字，连续地放入地址768、769，并调用770为入口的程序，便可以演奏一首乐曲了。

二、演奏一首乐曲的程序

有了以上基础，我们可以编写一个演奏一首乐曲的程序。除了现有的数据外，首先必须决定哪个数字对应于钢琴中的某个音键。下表中的A表示C调中的1（DO）。

	第一八音度	第二八音度
A	228	114
B	204	102
C	192	96
D	171	85
E	152	76
F	144	72
G	128	64

我们可以利用这些数字输入乐曲的各个音阶。在此同时，还要定出表示各音符持续时间（节拍）的固定数字。凭经验定出一些数值如下：

四分音符　　240

八分音符　　120

十六分音符　60

假定已选好某一首乐曲。先列出这首乐曲每一音符的两个数字（即音阶和节拍），并将它们寄存于DATA语句中。下面先给出这个程序，然后再加以解释。

```
10  HOME
20  DIM  P(60), D(60)
99  REM  MACHINE  CODE  PROGRAM
100  DATA  172, 1, 8, 174, 1, 8, 169, 4, 82,
     168, 252, 178, 48, 192, 282, 208, 258, 186,
     208, 289, 206, 0, 8
110  DATA  208, 281, 96, 82, 82, 112, 8, 24,
     216, 186, 8, 160, 160, 16, 56
120  FOR  N=770  TO  807
```

```
140 READ Y
160 POKE N, Y
180 NEXT N
199 REM DATA FOR PITCHES OF NOTES
200 DATA 171, 152, 128, 114, 102, 85, 102,
    114, 128, 152, 171, 171, 171
202 DATA 102, 96, 85, 76, 85, 102, 171, 152,
    128, 114, 128, 114, 102
204 DATA 102, 96, 85, 76, 85, 102, 171, 152,
    128, 114, 102, 114, 128
206 DATA 114, 102, 102, 114, 128, 114, 102,
    85, 102, 114, 128, 152, 171, 171, 171
219 REM DATA FOR DURATIONS OF
    NOTES
220 DATA 40, 40, 40, 240, 60, 60, 240, 60,
    60, 120, 180, 60, 240
222 DATA 60, 60, 240, 60, 60, 120, 120, 60,
    60, 240, 60, 60, 240
224 DATA 60, 60, 240, 60, 60, 120, 120, 120,
    120, 120, 40, 40, 40
226 DATA 120, 240, 60, 60, 180, 60, 60, 60,
    240, 60, 60, 120, 180, 60, 240
240 FOR C=1 TO 54
260 READ P(C):REM PITCHES
280 NEXT C
800 FOR C=1 TO 54
820 READ D(C):REM DURATIONS
```

```
340  NEXT  C
359  REM  ROUTINE  TO  PLAY  TUNE
360  FOR  C=1  TO  54
380  POKE  768, P(C)
400  POKE  769, D(C)
420  CALL  770
440  NEXT  C
```

在程序中，第200至206句包含了这首乐曲的音阶数据（因有54个音符，所以有54个数字）；而第220至226句包括了54个音符的节拍数据；第240～280句是把音阶数据存放在一个称为P的数组中；第300～340句是将节拍数据存放在一个称为D的数组中。当然，必须事先定义这两个数组（第20句），接着便可利用循环POKE把第一个音阶和第一个节拍存入768和769地址，再用CALL调用770入口的机器语言发音子程序发出第一个音。然后程序继续循环发出第二音，第三音，……，一直到全部54个音演奏完毕。

这个程序可以演奏任何乐曲。但是最好把200至226这几句写成两个子程序，以便存入磁盘或从磁盘取出，因为这些数据是随着每一首乐曲而变化的。

三、旋律产生器程序

下面以上述的音乐演奏程序为基础，介绍一个完整的“旋律产生器”程序。它可以从键盘输入一首乐曲，保存在磁盘上，并能演奏这首乐曲。

以下给出程序清单，供读者研究、使用。

```
100  DATA  172, 1, 3, 174, 1,3, 169,4,32,168,
     252,173, 48, 192, 232, 208, 253, 136, 208,
     239, 206, 0, 3, 208, 231, 96, 32, 32, 112,
```

```
    8, 24, 216, 186, 8, 160, 160, 16, 56
110 FOR X=770 TO 807
120 READ Y
130 POKE X, Y
140 NEXT X
300 DIM A(200), B(200)
400 HOME: PRINT"      WRITING MUSIC"
405 PRINT: PRINT"DO YOU WISH TO: "
407 PRINT: PRINT
410 PRINT" BEGIN A NEW TUNE    B"
415 PRINT" FINISH FOR NOW      F"
420 PRINT: PRINT" LIST NOTES ON
    SCREEN                 L"
425 PRINT" SAVE A TUNE ON DISK S"
430 PRINT: PRINT" RECOVER A TUNE
    FROM DISK              R"
432 PRINT" PLAY THE TUNE       P"
433 PRINT: PRINT" CHANGE A NOTE
    ON TUNE                C"
434 PRINT" ADD NEW NOTE TO TUNE A"
435 INPUT Y$: IF Y$="B" THEN 451
437 IF Y$="F" THEN 452
438 IF Y$="L" THEN 454
440 IF Y$="S" THEN 456
442 IF Y$="C" THEN 458
443 IF Y$="A" THEN 460
445 IF Y$="R" THEN 462
```

```
447 IF Y$="P" THEN 464
450 PRINT: PRINT"      TRY AGAIN":
    GOTO 400
451 GOSUB 500: GOTO 400
452 GOSUB 3000: GOTO 400
454 GOSUB 2500: GOTO 400
456 GOSUB 1500: GOTO 400
458 GOSUB 1700: GOTO 400
460 GOSUB 1800: GOTO 400
462 GOSUB 1600: GOTO 400
464 GOSUB 2000: GOTO 400
500 HOME: PRINT"      WRITING MUSIC":
    Z=0
501 PRINT
502 PRINT" **************"
505 PRINT"INPUT EACH NOTE AS A3-
    SYMBOL"
507 PRINT"COMBINATION AS SHOWN
    BELOW N STANDS"
509 PRINT"FOR NATURAL AND S FOR
    SHARP"
510 PRINT: PRINT" G G2N OR G2S"
512 PRINT"F——F2N OR F2S"
515 PRINT" E E2N"
517 PRINT"D——D2N OR D2S"
520 PRINT" C C2N OR C2S"
522 PRINT" B——B2N"
```

```
525 PRINT"  A  A2N  OR  A2S"
527 PRINT"  G——G1N  OR  G1S"
530 PRINT"  F  F1N  OR  F2S"
532 PRINT"  E——E1N"
535 PRINT"  D  D1N  OR  D1S"
537 PRINT"  C——C1N  OR  C1S MIDDLE C"
540 PRINT"  B  B1N"
545 PRINT"  A——A1N  OR  A1S"
550 PRINT: PRINT"  PRESS  A  KEY"
560 GET-Y $
565 PRINT: PRINT"  NOTE  THAT  EACH
    OCTAVE  RUNS  FROM  A  TO  G"
570 PRINT: PRINT"WHEN  YOU SEE THE
    QUESTION  MARK  INPUT"
575 PRINT" THE 3-LETTER COMBINATION
    FOR THE FIRST"
580 PRINT"NOTE  AND  PRESS  RETURN
    THEN  INPUT  THE"
585 PRINT"NUMBER, AS  REQUESTED,
    FOR  THE  DURATION  OF"
590 PRINT"THE  NOTE  AT  THE  END,
    TYPE  IN  END"
650 PRINT: PRINT: PRINT"NOW  INPUT
    A  NOTE  IN  THE  3-LETTER  FORM:"
652 IF Z$ = "END"THEN 1030
653 L$ = LEFT$ (Z$,1) : M$ = MID$ (Z$, 2,
    1) : R$ = RIGHT$ (Z$,1) : Z = Z + 1
```

```
655 IF X1>0 THEN Z1=Z:Z=X1
660 L=ASC(L$)-64
661 IF L>7 THEN 680:IF L<1 THEN 680
665 M=VAL(M$):IF M=2 THEN M=15:
    GOTO 670
666 IF M=1 THEN 670
667 GOTO 680
670 R=ASC(R$)
672 IF R=78 THEN R=1:GOTO 685
675 IF R=83 THEN R=8:GOTO 685
680 PRINT:PRINT "TRY AGAIN":GOTO 650
685 PRINT:PRINT:PRINT "NOW INPUT
    LENGTH OF NOTE:"
687 PRINT:PRINT"240 FOR A MINIM"
690 PRINT:PRINT"120 FOR A CROTCHET"
694 PRINT:PRINT"30 FOR A SEMI-QUAVER"
695 PRINT:PRINT:INPUT"   ";W:B(Z)=W
696 IF INT (W/15)=W/15 THEN 700
698 PRINT:PRINT:PRINT "TRY AGAIN":
    GOTO 695
700 N=L+M+R-2
710 ON N GOTO 750, 760, 770, 780, 790, 800,
    810, 820, 830, 840, 850, 860, 870, 880, 890,
    900, 910, 920, 930, 940, 950, 960, 970, 980,
    990, 1000, 1010, 1020
750 A(Z)=228:GOTO 650
760 A(Z)=204:GOTO 650
```

```
770  A(Z)=192:GOTO 650
780  A(Z)=171:GOTO 650
790  A(Z)=152:GOTO 650
800  A(Z)=144:GOTO 650
810  A(Z)=128:GOTO 650
820  A(Z)=216:GOTO 650
830  A(Z)=192:GOTO 650
840  A(Z)=181:GOTO 650
850  A(Z)=161:GOTO 650
860  A(Z)=144:GOTO 650
870  A(Z)=136:GOTO 650
880  A(Z)=121:GOTO 650
890  A(Z)=114:GOTO 650
900  A(Z)=102:GOTO 650
910  A(Z)=96:  GOTO 650
920  A(Z)=85:  GOTO 650
940  A(Z)=72:  GOTO 650
950  A(Z)=64:  GOTO 650
960  A(Z)=108:GOTO 650
970  A(Z)=96:  GOTO 650
980  A(Z)=90:  GOTO 650
990  A(Z)=80:  GOTO 650
1000 A(Z)=72:GOTO 650
1010 A(Z)=68:GOTO 650
1020 A(Z)=60:GOTO 650
1030 IF X1>0 THEN Z=Z1
1040 RETURN
```

```
1500 D$ =CHR$(4)
1502 PRINT : PRINT : INPUT " WHAT FILE
     NAME"; N$
1505 PRINT D$; "OPEN"; N$
1510 PRINT D$; "WRITE"; N$
1515 FOR J=1 TO Z
1520 PRINT A(J) : PRINT B(J)
1525 NEXT J
1530 PRINT D$; "CLOSE"; N$
1535 PRINT" NOW SAVED ON DISK PRESS
     ANY KEY"
1540 GET Y$
1550 RETURN
1600 INPUT "HOW MANY NOTES WERE
     SAVED:"; Z
1605 PRINT : PRINT : INPUT" WHAT FILE
     NAME" ; N$
1610 D$ =CHR$(4)
1615 PRINT D$;"OPEN";N$
1620 PRINT D$;"READ";N$
1625 FOR J=1 TO Z
1630 INPUT A(J) : INPUT B(J)
1635 NEXT J
1640 PRINT D$; "CLOSE"; N$
1645 PRINT"NOW RECOVERED FROM DISK
     PRESS ANY KEY"
1650 GET Y$
```

```
1660  RETURN
1700  HOME PRINT"WHAT IS THE NO OF
      THE NOTE"
1705  PRINT : INPUT"THAT YOU WANT TO
      CHANGE";X1
1710  PRINT : PRINT"REMEMBER TO INPUT
      END AFTER THIS"
1715  PRINT : PRINT"PRESS ANY KEY" : GET
      Y$
1720  GOTO 650
1800  HOME
1810  HOME : PRINT"THE NOTES WILL BE
      ADDED TO THE"
1820  PRINT : PRINT"END OF THE TUNE
      AS USUAL WHEN"
1830  PRINT : PRINT"YOU WISH TO STOP,
      TYPE IN END"
1840  GOTO 650
2000  FOR  J= 1 TO  Z
2010  POKE 768, A(J) : POKE 769, B(J) : CALL
      770
2020  NEXT  J
2030  RETURN
2500  HOME FOR  J= 1 TO  Z
2510  PRINT A(J), B(J)
2515  IF INT (J/20) = J/20 THEN 2517
2516  GOTO 2520
```

```
2517 PRINT "PRESS A KEY":GET Y$
2520 NEXT J
2530 PRINT:PRINT "PRESS ANY KEY":
     GET Y$:RETURN
3000 HOME:PRINT "THANK YOU FOR
     NOW":END
```

第四节　提高磁盘效率的技巧

磁盘效率通常包括磁盘空间的利用率及读／写磁盘的速度等方面。本节为读者介绍两个程序：一个是用来扩大数据磁盘的用户容量的"数据磁盘格式化程序"；另一个是"快速载入磁盘信息的程序"。

一、DOS 数据磁盘格式化程序

当我们使用 DOS 的 INIT 命令对磁盘初始化时，命令的处理过程分为以下四个步骤。

（1）格式化磁盘。将磁盘划分为＄0～＄22共35条磁道，每条磁道有＄0～＄F共16个扇区的格式。这样，整个磁盘上共有35×16＝560个扇区，每个扇区可以存放256个字节，磁盘的容量为560×256/1024＝140K字节。

（2）把＄11磁道的＄0扇区定义为册内容表，或称VTOC表（Volume Tabel Of Contants）。DOS称一个磁盘为一册(Volume)，VTOC表上记载了磁盘中每个扇区是"空"还是"被使用"的使用情况。将＄11磁道的其余扇区＄F～＄1定义为供磁盘文件目录使用，然后把磁道＄0～＄2及＄11在VTOC表中置为"被使用"。

（3）从内存中把DOS抄入＄0～＄2磁道。

(4) 将内存中的“欢迎”程序送入磁盘。

因此，在已初始化的磁盘中，DOS 占了3个磁道，VTOC表及文件目录共占 1 个磁道，还有31条磁道共124K字节空间供用户使用，扣除“欢迎”程序所占的空间后，实际剩下的空间就更少了。事实上，有时并不要求每个磁盘都可以“自动导入”，尤其是一些用于存放数据的磁盘。下面所提供的程序，运行后将修改内存中 DOS 的“INIT”命令程序，修改后的“INIT”命令只对一磁盘进行“格式化”，“格式化”后不写入DOS及“欢迎”程序，而将$1、$2磁道释放出来给用户使用，$0磁道由于有其它的原因仍不能使用。

DOS 数据磁盘格式化程序如下：

```
1   REM   * * * * * * * * * * * * * * * *
2   REM   *                             *
3   REM   *  DATA DISK FORMATOR         *
4   REM   *                             *
5   REM   * * * * * * * * * * * * * * * *
100  TEXT : HOME : VTAB  3
110  INVERSE : HTAB  8
120  PRINT "DOS  DATA  DISK
     FORMATOR"
130  NORMAL : PRINT
140  PRINT "BY  X. Y. Wang  of  ZhongShan
     University."
150  PRINT : PRINT
160  PRINT "THIS  PROGRAM  MODIFIES
     DOS  TO  INITIALIZE"
170  PRINT "A  DISKETTE  WITHOUT
```

```
    PUTTING DOS ON TRACK"
180 PRINT "1-2. THIS MEANS THAT
    YOU WILL HAVE AN"
190 PRINT "ADDITIONAL 8K OF SPACE
    FOR INFORMATION"
200 PRINT "STORAGE."
210 PRINT : PRINT
220 REM DON'T SAVE GREETING
    PROGTEM
230 POKE 42347, 96: REM RTS
240 REM SET VTOC, TRACK 0 USED
250 POKE 44723,4: REM SKIP TRACK 0
260 REM NOT WRITE DOS IN TRACK
    0-2
270 POKE 44799, 76: POKE 44800,127:
    POKE 44801, 179: REM JMP $B37F
280 INPUT "ENTER DISK VOLUME>";
    V$
290 IF V$=" " THEN V$="0"
300 V=VAL(V$)
310 IF V<0 OR V>254 THEN PRINT
    CHR$(7);: VTAB 15: GOTO 280
320 PRINT
330 PRINT "      INSERT DISKETTE TO
    BE INITIALIZED"
340 PRINT "          PRESS [RETURN]
    TO PROCEED"
```

```
350 PRINT "                    [ESC] TO
    EXIT"
360 GET K$
370 IF K$=CHR$(27)THEN 450
380 IF K$< >CHR$(13)THEN PRINT
    CHR$(7);: GOTO 360
390 NORMAL : PRINT
400 PRINT CHR$(4);"INIT XXX,V";V
410 PRINT "ANOTHER(Y/N)?";
420 GET K$: IF K$="N" THEN
    GOTO 450
430 IF K$< >"Y" THEN PRINT
    CHR$(7): VTAB 22: GOTO 410
440 GOTO 100
450 REM RETRIEVE DOS
460 POKE 42347, 76: POKE 44723, 12:
    POKE 44799, 32: POKE 44800, 194:
    POKE 44801, 183
999 PRINT : END
```

程序中主要语句的含义如下：

第230句令 INIT 命令处理程序，不把内存的程序作为“欢迎”程序存入磁盘；

第250句释放 VTOC 中的＄1和＄2磁道，但仍保留＄0磁道；

第270句使 INIT 命令处理程序，不把 DOS 抄入＄0～＄2磁道。

第460句恢复被修改的地址单元的数值。

二、DOS快速载入程序

在使用FID程序复制文件时，我们会发现它从磁盘中读取文件的速度，要比同一文件从磁盘中LOAD或BLOAD入内存时快得多，文件越大，这种差别就越明显。这是为什么呢？DOS读磁盘是通过其RWTS（read or write a track and sector），即“读或写磁道及扇区程序”进行的，但RWTS程序只能用于读或写一个满的扇区，而不能读那些只用了一部分的扇区。而大多数文件的最后一个扇区一般是不会恰好用完的。因此DOS从磁盘上读取文件时，先用RWTS将一个扇区的信息读入文件缓冲区，然后再复制到用户程序区去，这样就比直接从磁盘上读入用户程序区中的信息要花费更多的时间了。当然，影响速度的还有读完一个扇区之后等待磁盘旋转到下一个扇区的时间。

这个快速载入程序将代替原来DOS系统中的载入程序。当LOAD或BLOAD、RUN或BRUN一个文件时，该程序将文件的第一个扇区的信息读入文件缓冲区中，把文件的长度记录下来，并把该扇区抄到用户程序区里。如果载入程序根据文件长度算出还有多于256个字节要从文件中读出的话，就应把下一个扇区（256字节）直接读入内存用户程序区中。如此重复，直到测出文件中剩下少于256个字节为止（未用完的扇区）。最后再通过RWTS的缓冲区，将其余的字节从文件中读入内存用户程序空区。由于快速载入程序不需要通过文件缓冲区把每个扇区从磁盘中读出，故在读取文件时可以比原来DOS中的方式快4～5倍。快速载入程序约250字节，它占用了DOS“INIT”程序所在的位置。“INIT”命令则变为只能在已初始化的磁盘上，写入带有快速载入程序的DOS系统，并清除磁盘上的目录。快速载入程序可在监控状态下输

入计算机中，然后用INIT命令将它存入磁盘；或者用

BSAVE LOADER. OBJ, A$BEAF, L$ED

然后通过一条BASIC程序将其抄入磁盘中，这样就可以反复使用了。

注意，有些用机器语言编写的程序如游戏等，需要使用被快速载入程序占用空间中的原DOS机器语言子程序，因而这类程序不能用快速载入程序处理。

DOS快速载入程序如下所示：

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
BEA0—																A9
BEB0—	00	4C	43	BE	AD	C9	B5	8D	6C	BF	8D	75	BF	AD	CA	B5
BEC0—	8D	6D	BF	8D	76	BF	AD	CB	B5	8D	F8	BE	AD	CC	B5	8D
BED0—	F9	BE	EE	BD	B5	38	AD	C8	B5	ED	BD	B5	85	42	AD	C4
BEE0—	B5	E9	00	85	43	AC	BD	B5	AE	C1	B5	8A	D0	08	AD	C2
BEF0—	B5	F0	58	CE	C2	B5	CA	B9	00	96	91	42	C8	D0	EC	8C
BF00—	EB	B7	8E	C1	B5	A9	0E	8D	C6	B5	E6	43	AE	C2	B5	F0
BF10—	1A	A5	42	8D	BF	B5	A5	43	8D	C0	B5	20	49	BF	EE	C0
BF20—	B5	CE	C2	B5	D0	F5	AD	C0	B5	85	43	AD	CB	B5	8D	BF
BF30—	B5	AD	CC	B5	8D	C0	B5	AE	C1	B5	F0	0A	20	49	BF	AE
BF40—	C1	B5	A0	00	F0	A5	4C	EA	A2	AC	C6	B5	D0	0D	AD	C9
BF50—	B5	AE	CA	B5	A0	01	20	65	BF	A0	0C	AD	BF	B5	AE	C0
BF60—	B5	20	65	BF	60	8D	F0	B7	8E	F1	B7	B9	00	97	F0	18
BF70—	8D	EC	B7	C8	B9	00	97	8D	ED	B7	C8	8C	C6	B5	A9	B7
BF80—	A0	E8	20	B5	B7	B0	08	60	20	EA	A2	A2	05	D0	05	20
BF90—	EA	A2	A2	08	68	68	68	68	8A	4C	D2	A6				

```
1   REM      * * * * * * * * * * * * * *
2   REM      *                         *
             *                         *
3   REM      * DOS 3.3 FAST LOADER *
4   REM      *                         *
             *                         *
5   REM      * * * * * * * * * * * * * *
100 PRINT CHR$ (4); "BLOAD LOADER.
    OBJ, A$BEAF"
110 POKE 42103, 76
120 POKE 42104, 180
130 POKE 42105, 190
150 TEXT : HOME
160 VTAB 6 : HTAB 6 : INVERSE
170 PRINT "-DOS QUICK LOADER
    BUILDER-"
180 NORMAL : PRINT : HTAB 5
190 PRINT "BY X.Y.WANG FROM BYTE
    NOV, 1988"
200 PRINT
210 PRINT "THIS PROGRAM BUILDS A
    QUICK LOADER"
220 PRINT "ROUTINE FOR DOS WHICH
    CAN LOAD PROGRAMS"
230 PRINT " FIVE TIMES FASTER THAN
    NORMAL DOS DOES."
240 PRINT
250 PRINT "USE THE' INIT' COMMAND
    TO PUT FAST"
```

```
260 PRINT "DOS IN AN";
270 INVERSE : PRINT"INITIALIZED",
280 NORMAL : PRINT "DISKETTE."
290 PRINT
300 PRINT "NOTE :"; : INVERSE
310 PRINT "ANY DATA ON THE
    DISKETTE TO BE"
320 HTAB 7: PRINT "INITIALIZED WILL
    BE DESTORYED! "
330 NORMAL:PRINT :END
```

附录一 FP BASIC 语言（APPLESOFT BASIC）摘要

1. 简单变量

类　　型	名　　称	范　　围
实　　数	AB	±9.99999999E+37
整　　数	AB%	±32767
字　　串	AB$	0至255个字符

其中A为大写英文字母（A，B，…，Z），B为大写英文字母或数字（0，1，…，9）。变量名称可多于两个字符，但只有前面两个字符有效。例：AB%和AB3QS%被看作是相同的整数变量。

2. 数组变量

类　　型	典型元素的名称
实　　数	AB（3，12，7）
整　　数	AB%（3，12，7）
字　　串	AB$（3，12，7）

数组的大小受到用户RAM（内存）的限制。

3. 代数运算符号

= 给变量赋值（LET可有可无）

− 负数

∧或↑ 指数运算

* 乘法运算

/ 除法运算

\+ 加法运算

- 减法运算

4. 关系与逻辑运算符

= 等于

<> 不等于

< 小于

> 大于

<= 小于或等于

>= 大于或等于

NOT 逻辑“非”

AND 逻辑“与”

OR 逻辑“或”

关系与逻辑运算式，若结果为真，其值为1。若结果为假，其值为0。关系运算符亦可用于字符串的比较。

5. 系统和应用程序命令

命　　令	功　　能
LOAD	将卡式录音带上的程序取入内存（必须在执行LOAD命令之前，先将磁带转到程序开头的地方，且按下录音机上的PLAY键。）。当LOAD的动作开始和结束时，APPLE Ⅱ机都将发出"Bi"的一声。只有按RESET键才能打断这一动作
SAVE	将内存中的程序存到卡式录音带上（必须在执行SAVE命令之前，先对准一段空白磁带的开头，且按下RECORD及PLAY键。）由于SAVE命令不会检查用户按的键是否正确，因此用户必须正确操作。当开始存入及存完的时候，用户都会听到"Bi"的一声
NEW	清除目前内存中的程序。执行了NEW命令，内存中将没有任何BASIC程序
RUN	由最低的行号开始执行程序
RUN 行号	范例：RUN 250 由行号250开始执行程序（在执行RUN之前，先把所有变数X，Y，A及堆栈存储器清掉）
STOP	暂停执行程序，并指示停止的行号
END	停止执行程序，但不会有任何信息印出。用于结束程序的操作
CTRL-C	用于立即命令（模式），停止执行程序或列表，并回到目前的操作状态
CONT	继续执行由STOP、END、CTRL-C或CTRL RESET命令所停止的程序

（续上表）

命　令	功　能
TRACE	辅助侦错，使得每一个正在被执行的语句的行号显示在屏幕上。TRACE命令不能被RUN、CLEAR、NEW、DEL或RESET取消，它只能被NOTRACE命令取消
NOTRACE	关闭（取消）侦错状态
PEEK(X)	取出内存单元X中的内容（X为十进数地址，它的内容也是十进数）。 范例：B = PEEK(37)就是取出地址37的内容，并赋值给变量B
POKE X, 数值	把指定的数值存入指定的地址X（X为十进数地址，数值也是十进数）。 范例：POKE4096，05就是把数05存入地址单元4096内
WAIT X, Y, Z	条件等待——暂停程序的执行，直到内存单元X的内容与数值Z相“异或”（EOR），再与数值Z相“与”（AND）后得到非零的结果才继续执行程序。假定我们要测试内存单元X中的某些位是1或0，那么第二个数值Y的二进制值就表示你要测试的是哪一位，其中1表示要测的位，0表示不必测试的位。至于第三个数值Z不是0就是1，它表示你希望测试的位的情况：取1表示被测位必须是0，取0表示被测位必须是1，程序才能继续执行下去（否则将由WAIT暂停等待）。若你没指定第三个数值Z，则机器自动假定它是0。 范例：WAIT　1320，255 或　WAIT　1320，255，0

（续上表）

命　　令	功　　　　能
CALL　X	执行（调用）起始于内存单元X的机器语言子程序（X为十进数地址单元）。 范例：CALL-922 执行入口地址为－922的子程序(由－922入口的子程序在屏幕上产生跳行的动作）
USR(N)	将N值传送至机器语言子程序中。N值将会被放到浮点累加器（地址＄9D到＄A8），再执行一个JSR指令（转子程序，入口在＄0A），若用户在＄0A至＄0C内放一条JMP指令，机器语言子程序便可由JMP去执行。由子程序计算得到的值也是被放到浮点累加器内的。 范例：＊　0A:4C　68　0B ＊　0B68:60 ＊　CTRL-C]　PRINT USR　(6)＊3 18] 在＄0A起存放JMP＄0B68　（跳到＄0B68） 在＄0B68存放RTS(返回)当执行到 USR(6)时，参数6被放进浮点累加器中，而监控程序执行指令JSR到地址＄0A时发现JMP要跳到＄0B68，而＄0B68是RTS，便需要返回原来的 APPLESOFT。而函数USR的返回值也就是以前放在累加器的6；于是APPLESOFT将其乘以3得18，并打印出来
HIMEM	设定在APPLESOFT程序中，可以使用的最高内存单元。它被用作保护存贮的数据、高分辨图形或机器语言的存储单元。这命令不会被CLEAR，RUN，NEW，DEL等指令及RESET改变。 范例：HIMEM：16384
LOMEM	设定APPLESOFT程序所能使用的最低地址单元，用以保护变量不被高分辨图形所损害。 范例：LOMEM：2060但打入NEW会改变LOMEM的值

6. 与编辑和格式相关的命令

命令	功能
LIST	列出整个程序（从最低行号起）
LIST X-Y 或LIST X,Y	列出由行号X到Y的程序语句。 范例：LIST 100，150
DEL X，Y	删除由行号X到Y的程序　（Y>=X） 范例：DEL 200，215
REM	程序注释：REM 是一种非执行的语句。注释内容可以任意。 范例：REM THIS A CONT PRG
VTAB Y	将屏上光标垂直移到第Y行（Y 在1至24之间取值），顶行为1，底行为24。 范例：VTAB 15
HTAB X	将光标水平移到目前所在行的第X位置（X在1至40之内取值）。 范例：HTAB 23
TAB（X）	从目前光标所在行的最左边算起，将光标移到第X位置（X在0至255之间取值，0至39在本行上，而40～79在下一行上，以此类推）。TAB（X）仅用在 PRINT 语句中 范例：PRINT TAB（23）；"*"
POS（0）	读得（传回）目前光标的水平位置（0至39）。放在括号内的值并没有任何关系，只要它能被APPLESOFT接受就可以了

（续上表）

命　令	功　能
SPC（X）	在目前光标位置的前项与欲印的下一项之间放置X个空格，留空X个空格。X在0至255之间取值。SPC(X)仅用在PRINT语句中，SPC(X)后面要有分号及打印项 范例：PRINT SPC (8); "*" PRINT TAB(3); "*"; PRINT SPC (8); "*"
HOME	清除屏幕目前的信息，并将光标移至屏的左上角，但不清除内存
CLEAR	把所有已赋值的变量清除为零，将所有的字符串清除掉，并将堆栈寄存器清除干净
FRE（0）	告诉用户还有多少个内存单元可用（以字节为单位）。括号内的值无任何意义，只要APPLESOFT可接受即可。这命令同时将不再被使用的字符串从内存中清除掉。 范例：B＝FRE（0） PRINT B
FLASH	将屏幕输出设定为闪烁状态
INVERSE	将屏幕输出设定为白底黑字的反转状态
NORMAL	关闭闪烁或反转状态，以正常状态显示
SPEED＝X	设定字符输出至屏幕或其它I/O设备的速度，最低速度是X＝0，最高速度是X＝255。 范例：SPEED＝50
〔ESC〕A或K	将光标向右移一格

（续上表）

命　　令	功　　　　能
〔ESC〕B或J	将光标向左移一格
〔ESC〕C或M	将光标向下移一格
〔ESC〕D或I	将光标向上移一格
→	将光标所在位置的字符输入内存，并将光标往右移一格
←	将光标左移一格，并把经过的字符从内存中删除
CTRL-X	删除目前键盘输入的语句行，返回原来的语句行

7. 与数组和字符串相关的命令

命　　令	功　　　　能
DIM A（X,Y,Z）	设定数组A所标注的范围，预留（X+1）*（Y+1）*（Z+1）个实数元素的内存空间，供这个数组使用。数组以A（0，0，0）开始。 范例：DIM AG（3，4）
DIM A$（X，Y）	设定字符串数组A$的标注范围包含X+1*Y+1个字符串元素，每个元素最长为255个字符。 范例1：DIM GRH$（5，10） 范例2：DIM AGE(20，3)，NAME$（50） 范例2中，NAME$（50）就可以容纳51个（由0至255个字符长度的）字符串；AGE（20，3）则可以存21×4＝84个实数。 如果用户在程序中并没有使用DIM命令定义一个数组，那么该数组最多只能使用11个元素（自己设定值为10）。

（续上表）

命　令	功　能
DIM A$（X,Y）	当用户执行RUN或是CLEAR命令时，数组内所有元素的值都被清为零
LEN(A$)	计算字符串A$的长度（最多不可超过255）。 范例：？LEN（"AN APPLE A DAY"）或 A$＝"AN APPLE A DAY"；？LEN(A$)
STR$(X)	将元素X转换成相应的字符串。 范例：STR$（12·45）将得到字串"12·45"
VAL(A$)	将一个字串A$转变成为一个实数或整数值。第一个非数值字符以前的字符当作数值送回。而出现第一个非数值字符之后，均作零来处理。 范例：VAL("－3.7E4A5PLE") 得到的值就是—37000
CHR$(X)	求得X值所对应的ASCII字符（X是该字符的ASCII码，X值在0与255之间）。 范例：CHR$（65） 该字符函数求得A字（因为A的ASCII码为65）
ASC(A$)	求得字符串A$中第一个字符的ASCII码。 范例：ASC（"QUEST"）可得到81（因为Q的ASCII码为81）
LEFT$（A$,X）	取出字符串A$的最左边的X个字符。 范例：A$＝"APPLESOFT"；PRINT LEFT$（A$，5） 将得到APPLE（从左取的5个字符）

（续上表）

命　　令	功　　　　　　　　　　能
RIGHT＄（A＄,X）	取出字符串 A＄最右边的X个字符。 范例：B＄＝"LMICROSOFT"：PRINT RIGHT＄（B＄，4） 将得到SOFT（从右取的4个字符）
MID＄（A＄,X,Y）	取出字符串A＄中从第X个字符开始的Y个字符。 范例：C＄＝"AN APPLE Ⅱ PLUS"：PRINT　MID＄（C＄，4，8） 将取出APPLE　Ⅱ六个字。
STORE　AG	将数值数组 AG 存于卡式磁带中（该命令不能直接用来储存字符串数组）。在执行STORE命令之前，要先按下RECORD及PLAY键。 范例：STORE　GF（7） 执行STORE 命令，便把 GF（0）、GF（1）、GF（2）、…、GF（7）都存入磁带内
RECALL　B	自卡式录音带上读回数组 B。数组 B 的维数必须设定与储存（STORE）时一致。也即是说，一个数组可以使用不同的名字 RECALL 它，但是必须在程序内使用DIM命令定义它。 在执行 RECALL 之前，必须先按下录音机的PLAY键。 范例：RECALL　GF（7） 　或　RECALL　QT（7） 存于磁带中的元素被读入 GF（0）、GF（1）、GF（2）、…、GF（7），或读入 QT（0）、QT（1）、QT（2）、…、QT（7）

8. 输入/输出命令

命 令	功 能
INPUT A$	在屏幕上打出问号"?"，并等待用户键入字符串A$的值
INPUT "NAME";A	在屏上印出NAME，并等待用户键入变量A的实数值
GET A$	等待用户键入一个字符给A$，而不必按RETURN键
DATA X, "Y", Z	建立数据的串列，以供READ语句使用。 范例：DATA "APPLE Ⅱ PLUS","DOS 3.3", 1.394, -13 第一个数据是字符"APPLE Ⅱ PLUS"; 第二个数据是字符串"DOS 3.3"; 第三个数据是实数1.394; 第四个数据是整数-13
READ A$	把DATA语句提供的数据读入READ语句的元素内 范例：READ A, B%, C$ DATA语句的前两个数据必须是数字，而第三个必须是字串，它们分别被读入READ中的A,B%,C$内
RESTORE	把读DATA语句的指针复原，使下一个READ语句再从该DATA语句的第一个数据读起。
PRINT "X=";X	在屏上印出字符串X=及变量X的值。分号用于紧凑连接印出的项目。 范例：PRINT "A$=";X PRINT "NAME1=";Z

（续上表）

命　　令	功　　能
IN#n	表示输入从SLOT#n所在的外围设备来的信息（其中，n=1，2，…，7）。 范例：IN#6
PR#n	将输出的信息送到SLOT#n所在的外围设备（其中，n=1，2，…，7）。 范例：PR#1
LET A=X	将X值赋给变量A。LET可省略。
DEF FN A(W)=数值表达式	用户自己定义函数A为指定的数值表达式，变量为W。用户先给出此定义，以后在程序的其它地方就可以使用这个函数了。 范例：DEF FN A(W)=2*W+W

9. 与程序流程和控制相关的命令

命　　令	功　　能
GOTO 语句行号	无条件地转去执行由GOTO语句指示的行号开始的程序
IF 关系表达式 THEN 语句行号	若关系表达式为真，则转去执行THEN指示的语句；若关系表达式为假，则执行IF语句的下一行语句
FOR X=e_1 TO e_2 STEP e_3 循环体 NEXT X	这是FOR-NEXT语句，用此语句可以实现指定次数的循环： （1）执行FOR语句，先把初值e_1赋给循环变量X，并将终值e_2及步长e_3记下来，以便以后作比较用。 （2）执行FOR语句和NEXT语句之间的循环体。

（续上表）

命　　令	功　　　　能
FOR X = e_1 TO e_2 STEP e_3 循环体 NEXT X	（3）执行 NEXT 语句，循环变量X增加一个e_3值，即$X = X + e_3$。 （4）将已增值的循环变量 X与终值 e_2 比较，如 X 未超过终值 e_2，则再执行一次循环体和 NEXT 语句，即重复执行（2）、（3）、（4），直到 X 超过 e_2 为止。以后执行 NEXT 语句下面的一个语句
GOSUB 语句行号	转去执行由GOSUB指定行号的子程序
RETURN	指示子程序的结束，返回目前执行的 GOSUB语句的下一个语句
POP	从堆栈返回（RETURN）地址的 上方，取出一个返回地址。因此，在执行 POP 后，一个被 执行的RETURN命令会使程序的执行转移到第2个GOSUB命令的下一个语句去
ON表达式 GOTO行号1，行号2，…，行号n	跳到由表达式指示的第几个行号。 范例：ON　M　GOTO 100，200，300，905，1090 若M＝1，则执行GOTO 100；若M＝2，则执行 GOTO 200；若 M＝5，则执行 GOTO 1090。如果表达式的值是零或超过行号的数目，则计算机会执行下一个语句
ON M GO-SUB 行号1，行号2，…，行号n	与ON　M　GOTO 命令同类。这个命令的功能是依照表达式M的数值而执行一个 GOSUB命令
ONERR GOTO 语句行号	程序执行过程中发生错误时，跳到GOTO指示的行号，而不印出错误信息，也不停止执行程序。这是一种无错误处理方法，用来避免使程序停止执行的错误

（续上表）

命　令	功　能
RESUME	返回当初发生错误的语句。这是错误处理常规中的语句，在处理错误的程序段（参见 ONERR GOTO）结束的时候，使程序继续由发生错误的语句开始执行

10. 与图形和游戏控制相关的命令

命　令	功　能
GR	将屏幕设定为低分辨图形方式，屏幕上部为40×40的图形区域，底部 4 行为程序正文所用
COLOR = n	设定绘图颜色（n 在 0 到 15之间取值）。 范例：COLOR = 7
PLOT X,Y	在水平坐标 X及垂直坐标 Y处，绘出一个色点。X 及 Y 的范围为 0 ～39。而原点（ 0， 0 ）在屏幕左上角
HLIN X1, X2 AT Y	从点（X1，Y）到点（X2，Y）绘出一条水平线
VLIN Y1, Y2 AT X	从点（X，Y1）到点（X，Y2）绘出一条垂直线
SCRN（X，Y）	取得屏幕上点（X，Y）的颜色
HGR	将屏幕首页设定为高分辨率图形状态，即屏幕上部为280×160图形区域，底部 4 行为程序正文所用
HGR2	设定屏幕次页为高分辨率图形状态，使屏幕变为280×192图形区域，清除全屏为黑色，不留底部为程序正文所用 4 个显示行

（续上表）

命　　令	功　　　　能
HCOLOR =n	设定绘图颜色（n 从0～7），打入NEW其值不变
HPLOT X，Y	在水平坐标X及垂直坐标Y处，画出一个色点。X的范围为0～279；Y的范围为0～159（对HGR）或0～191（对HGR2）；原点（0，0）在屏幕左上角。 范例：HPLOT　100，90
HPLOT X1，Y1 TO X2，Y2	从点（X1，Y1）到点（X2，Y2）画出一直线，命令可以延伸到更多的点（Xn，Yn）。 范例：HPLOT 40，50 TO 80，90 TO 130，150 TO 40，50 便画出三段互相连接的直线
SHLOAD	从卡式磁带上读取形状表载入内存
DRAW S AT X,Y	以HCOLOR设定的颜色，以X,Y为起始位置，绘出形状表中定义为第S号的形状。在执行DRAW之前，COLOR，ROT及SCALE的值必须已被定义。 范例：DRAW　3　AT 80，100
XDRAW S AT X，Y	以屏幕上每一点原来颜色的互补色，绘出形状表中定义为第S号的形状。这使得用户能够轻易地将DRAW命令绘出的图形，使用XDRAW命令清除掉，而不至影响到底色。 范例：XDRAW　3　AT 100，120

（续上表）

命　　令	功　　　　能
ROT＝n	设定由DRAW或XDRAW所绘图形的旋转度。ROT＝0，使图形绘在原先定义的位置上；ROT＝16，使图形往顺时针的方向旋转90°。至于360°的旋转可以由ROT＝64完成
SCALE＝n	设定由DRAW或XDRAW所绘图形的尺寸比例。SCALE＝1，使图形完全依照形状表而绘出；SCALE＝255，则使图形扩大255倍（注意：SCALE＝0是图形扩大的极限，即扩大256倍）。 范例：SCALE＝50
PDL（X）	取得指定的游戏控制器X（由0号到3号）的现值（在0到255之间）。 范例：PDL（3）　取第3号控制器的值
PEEK（X—16287）	取得一个数值，若此值大于127，则表示游戏控制器X（由0号到2号）的按钮已被按下
PEEK（—16336）	使扬声器发出“Bi”声

11. 数字函数

命　　令	功　　　　能
SIN（X）	X弧度的正弦函数值
COS（X）	X弧度的余弦函数值
TAN（X）	X弧度的正切函数值
ATA（X）	X的反正切函数值（以弧度为单位）

（续上表）

命　　令	功　　　　能
INT（X）	取得小于或等于X的最大整数值。 范例：INT（2.39）=2 INT（-45.1234）=-46 INT（2.62）=2 INT（2.62+.5）=3 若要求四舍五入，则取INT（X+0.5）
RND（1）	取得在0与0.999999999之间的随机实数值
RND（0）	再次取得上次刚产生的随机数
RND(-3)	当用户在括号内使用负参数时，只要每一次负参数相同，那么产生的随机数是相等的，并且使随后采用正参数的RND命令依照一定的顺序产生随机数。只要用户在使用RND命令前没有使用负参数，那么每一次产生的随机数都是介于0与1之间的新数目
SGN（X）	符号函数。若X<0，则函数值为-1；若X=0，函数值为0；若X>0，函数值为1
ABS（X）	取得X的绝对值。 范例：ABS（-3.14159）　取得3.14159
SQR（X）	取得X的平方根值。 范例：SQR(2)　取得值为1.41421356
EXP（X）	取得以e（2.718289）为底，X为指数的函数值（即e^x）。 范例：EXP（-3.2）
LOG（X）	取得X的自然对数值（即lnX）。 范例：LOG（13）

关于磁盘功能请参阅DOS部分的内容。在APPLESOFT BASIC语言中，使用DOS命令之前，必须加上CTRL-D或CHR$（4）。

附录二　FP BASIC 语言的错误信息

1.BASIC错误信息及处理方法

如果在BASIC命令或程序的执行过程中发生错误，BASIC会在屏幕（TEXT）上显示出错误信息，并立即回到命令状态。这时，变量值及程序保持不变，但程序不能继续执行，而且所有的GOSUB和FOR循环计数器被置0。

为了避免因程序发生错误而中断执行，可以加入ONERR GOTO命令。用这个命令连接一条错误处理程序。

2.BASIC错误信息的格式

命令错误：? ×× ERROR

程序语句错误：? ×× ERROR IN XY

其中"××"表示错误的名字，"XY"代表语句的行号。在立即命令中发生错误时，出错信息中无行号。

3.各种错误信息及其含义和原因分析

1）CAN'T CONTINUE （不能继续执行）

当要执行一条不存在的程序，或程序出错后已增加或删除了某些语句，而仍用"CONT"命令时，则显示此错误信息。

2）DIVISION BY ZERO （除数为零）

表达式中除数为零。

3）ILLEGAL DIRECT （非法立即命令）

用INPUT，DEFIN或DATA语句作为命令。

4）ILLEGAL QUANTITY （非法值）

函数或字符串的参数超过范围的原因可能是：数组变量下标为负数；LOG，SQR函数中用了零或负数；乘方A∧B中，A是负数，B不是整数；在MID $、LEFT $、RIGHT

$、WAIT、PEEK、POKE、TAB、SPC、ON…GOTO或任何图形函数中用了不适合的数。

5） NEXT WITHOUT FOR （NEXT 无 FOR 对应）

NEXT中的循环控制变量名和目前FOR描叙的不一致，或只有NEXT语句但无FOR语句。

6） OUT OF DATA （数据完）

在执行READ语句时，DATA语句所描叙的数据区已被读完了。

7） OUT OF MEMORY （内存用完）

内存用完可能由以下原因引起：程序太长；变量过多；FOR-NEXT循环超过10层；GOSUB嵌套超过24层；表达式过于复杂；括号层次超过36层；设置的LOMEN：太高或太低；HIMEN：太低。

8） FORMULA TOO COMPLEX （公式太繁）

执行IF "××" THEN形式的语句超过两次。

9） OVERFLOW （溢出）

计算结果超过BASIC允许的数值范围而溢出。如果值太小，则被认为是零，不会出错。

10） REDIM'D ARRAY （重复定义数组）

在程序中，对同一数组作不同的定义。如前面已有DIM A（10），后面又有DIM A（100），中间无CLEAR语句，则出此错误。

11） RETURN WITHOUT GOSUB

执行到的RETURN 无 GOSUB对应。

12） STRING TOO LONG （字串太长）

在用"+"号将字符合并时，字符长度超过255个字符。

13） BAD SUBSCRIPT （下标非法）

使用的数组下标不符合该数组定义的范围。

14） SYNTAX ERROR （语法错）

由于命令拼错；括号不配对；……等错误，均出现此信息。

15） UNDEF'D STATEMENT （行号未定义）

在GOTO,GOSUB,THEN语句中使用了不存在的行号。

16） UNDEF'D FUNCTION （函数未定义）

使用了尚未被自定义函数语句DEF定义的函数。

17） TYPE MISMATCH （形式不符）

出现了配合不当的形式。例如，将字串赋给数字变量或将数字赋给字串变量；又如在函数中字串为数字或数字为字串等。

附录三　DOS错误信息

FP　BASIC语言的错误信息，是用问号（?）开头的。

例：? SYNTAX　ERROR

整数BASIC语言的错误信息，是以三个星号（* * *）开头的。

例：* * *SYNTAX　ERROR

而DOS的错误信息，则没有任何符号开头。

例：SYNTAX　ERROR

DOS的错误信息及其含义、原因或处理方法如下：

1）DISK　FULL　（错误代码：9）

表示磁盘已满，不能再存放（SAVE）文件资料了。用户可换上另一磁盘，或把该磁盘中的文件清掉(DELETE)，然后再存入文件资料。

2）END　OF　DATA　（错误代码：5）

表示数据已完，不能再读出数据了。当READ命令读到0时，也会出现此信息。

3）FILE　LOCKED　（错误代码：10）

表示文件已被锁住。当用户在已锁的文件上使用SAVE、BSAVE、WRITE、DELETE、RENAME命令时，则显示此错误信息。这时，用户可更换磁盘或将该文件解锁（UNLOCK）。

4）FILE　NOT　FOUND（错误代码：6）

表示找不到文件。这是指用户要找的文件在目录（CATALOG）中不存在。其原因可能是：①拼错文件名；②该文件在另一磁盘中；③该文件名已被清掉(DELETE)。

5） FILE TYPE MISMATCH （错误代码：13）

表示文件类型不相配。这可能是用户使用的DOS命令与该文件的类型不相配，或用户用错文件名。

注意：LOAD、RUN、SAVE命令只能用于浮点BASIC和整数BASIC程序中；CHAIN命令只能用于整数BASIC程序中；OPEN、READ、WRITE、APPEND、POSITION、EXEC命令只能用于文本文件中；BLOAD、BRUN、BSAVE命令只能用在机器语言子程序中。

6） I/O ERROR （错误代码：8）

表示接口出错。这是指使用磁盘取、存资料不成功。其原因可能是：磁盘机的门未关；磁盘机上无磁盘；所用磁盘未经初始化；磁盘插入不正确或所用的磁盘已损坏。

7） LANGUAGE NOT AVAILABLE （错误代码：1）

表示无此套语言。例如当用户想用整数BASIC语言，而这套语言不在ROM或磁盘时就会出现此错误。这时，用户可关掉主机，把该语言的接口卡插到相应的位置上再开机。若仍然不行，则可能是接触不良，这时用户得关机再操作一次。

8） NO BUFFERS AVAILABLE （错误代码：12）

表示无缓冲区可用。当所有的缓冲区都在使用中，而用户的DOS命令还需要另一缓冲区时，则会出现此信息。

9） NOT DIRECT COMMAND （错误代码：15）

表示不是直接命令。当把只能用于间接的命令用于直接命令时，会出现此信息。只能用于间接命令的DOS命令有：APPEND，OPEN，POSITION，READ，WRITE。

10） PROGRAM TOO LARGE （错误代码：14）

表示程序太长。其原因可能是：用户把一个 APPLE 机内存不能整个容纳的磁盘文件放入APPLE 内存；HIMEM：定得太低，使得内存太小；指针乱了。此时，用户可重做一次；如仍不行，则可改变 HIMEM：的值，以释放缓冲区。

11） RANGE ERROR （错误代码：2 或 3 ）

表示范围出错。当用户的DOS 命令的参数或 DOS 命令中的量超出了规定的值，就会出现此错误信息。

12） SYNTAX ERROR （错误代码：11）

表示语法出错。这可能是某个 DOS 命令中的字母，标点有错或次序不对。这时用户可按 RESET 键，或重新调入DOS。

13） VOLUME MISMATCH （错误代码：7 ）

表示磁盘册编号不对。当DOS命令中的V 参数值与初始化时所指定的磁盘记录册编号不相符时，就会出现此错误。此时，用户可用目录（CATALOG）中显示的册编号，来更正所用的册编号。

14） WRITE PROTECTED （错误代码：4 ）

表示已被“写”保护。当在一个已被保护的磁盘上使用SAVE、BSAVE、WRITE命令或指针被打乱，都会出现此信息。这时，可把磁盘右边缺口的胶纸撕开，再重新做一次。

附录四　6502的指令系统

微处理器（CPU）6502有56条基本指令，13种寻址方法。在整个指令系统中，有的指令只能用一种寻址方法，而有的指令最多可以用8种寻址方法，所以6502指令可扩展成151种具体的操作指令。

6502的指令系统如表一所示。从表中可以看出，某一条指令只能用13种寻址方法中的1种或最多8种。

表一　　6502 指 令 系 统

记忆符	意　　义	寻址方法	操作码	字节	影响标志
ADC	A+M+C→A,C	IMM	69	2	NZCV
		Z	65	2	
		Z, X	75	2	
		ABS	6D	3	
		ABS, X	7D	3	
		ABS, Y	79	3	
		(Z, X)	61	2	
		(Z), Y	71	2	
AND	A∧M→A (与)	IMM	29	2	NZ
		Z	25	2	
		Z, X	35	2	
		ABS	2D	3	
		ABS, X	3D	3	
		ABS, Y	39	3	
		(Z, X)	21	2	
		(Z), Y	31	2	
ASL	A或M左移一位	A	0A	1	NZC
		Z	06	2	
		Z, X	16	2	
		ABS	0E	3	
		ABS, X	1E	3	
BCC	非进位转移	REL	90	2	
BCS	有进位转移	REL	B0	2	
BEQ	零转移	REL	F0	2	

（续上表）

记忆符	意　　义	寻址方法	操作码	字节	影响标志
BIT	A∧M（与） $M_7 \to N$ $M_6 \to V$	Z ABS	24 2C	2 3	NZV
BMI	负转移	REL	30	2	
BNE	非零转移	REL	D0	2	
BPL	正转移	REL	10	2	
BRK	中止	IMP	00	1	I=1
BVC	非溢出转移	REL	50	2	
BVS	溢出转移	REL	70	2	
CLC	清进位	IMP	18	1	C=0
CLD	清标志D	IMP	D8	1	D=0
CLI	清标志I	IMP	58	1	I=0
CLV	清标志V	IMP	B8	1	V=0
CMP	A—M A与M比较	IMM Z Z，X ABS ABS，X ABS，Y （Z，X） （Z），Y	C9 C5 D5 CD DD D9 C1 D1	2 2 2 3 3 3 2 2	NZC
CPX	X—M X与M比较	IMM Z ABS	E0 E4 EC	2 2 3	NZC

（续上表）

记忆符	意　　义	寻址方法	操作码	字节	影响标志
CPY	Y—M Y与M比较	IMM Z ABS	C0 C4 CC	2 2 3	NZC
DEC	M—1→M	Z Z，X ABS ABS,X	C6 D6 CE DE	2 2 3 3	NZ
DEX	X—1→X	IMP	CA	1	NZ
DEY	Y—1→Y	IMP	88	1	NZ
EOR	A⊕M→A （异或）	IMM Z Z，X ABS ABS,X ABS,Y (Z,X) (Z),Y	49 45 55 4D 5D 59 41 51	2 2 2 3 3 3 2 2	NZ
INC	M+1→M	Z Z，X ABS. ABS，X	E6 F6 EE FE	2 2 3 3	NZ
INX	X+1→X	IMP	E8	1	NZ
INY	Y+1→Y	IMP	C8	1	NZ
JMP	无条件转移	ABS (ABS)	4C 6C	3 3	

（续上表）

记忆符	意　　义	寻址方法	操作码	字节	影响标志
JSR	转　　子	ABS	20	3	[illegible]
LDA	M→A	IMM	A9	2	NZ
		Z	A5	2	
		Z,X	B5	2	
		ABS	AD	3	
		ABS,X	BD	3	
		ABS,Y	B9	3	
		(Z,X)	A1	2	
		(Z),Y	B1	2	
LDX	M→X	IMM	A2	2	NZ
		Z	A6	2	
		Z,Y	B6	2	
		ABS	AE	3	
		ABS,Y	BE	3	
LDY	M→Y	IMM	A0	2	NZ
		Z	A4	2	
		Z,X	B4	2	
		ABS	AC	3	
		ABS,X	BC	3	
LSR	A或M右移一位	A	4A	1	ZC N＝0
		Z	46	2	
		Z,X	56	2	
		ABS	4E	3	
		ABS,X	5E	3	
NOP	空操作	IMP	EA	1	

（续上表）

记忆符	意　义	寻址方法	操作码	字节	影响标志
ORA	AVM→A（或）	IMM	09	2	NZ
		Z	05	2	
		Z, X	15	2	
		ABS	0D	3	
		ABS, X	1D	3	
		ABS, Y	19	3	
		(Z, X)	01	2	
		(Z), Y	11	2	
PHA	A入栈	IMP	48	1	
PHP	标志入栈	IMP	08	1	
PLA	A出栈	IMP	68	1	N Z
PLP	标志出栈	IMP	28	1	由栈内容定
ROL	A或M循环左移一位	A	2A	1	NZC
		Z	26	2	
		Z, X	36	2	
		ABS	2E	3	
		ABS, X	3E	3	
ROR	A或M循环右移一位	A	6A	1	NZC
		Z	66	2	
		Z, X	76	2	
		ABS	6E	3	
		ABS, X	7E	3	
RTI	从中断返回 P出栈 PC出栈	IMP	40	1	由栈内容定

（续上表）

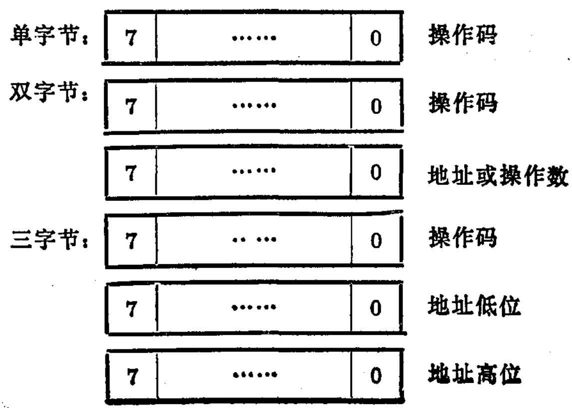
记忆符	意　　义	寻址方法	操作码	字节	影响标志
RTS	从子程序返回	IMP	60	1	
SBC	$A-M-\overline{C}\rightarrow A$	IMM	E9	2	NZCV
		Z	E5	2	
		Z,X	F5	2	
		ABS	ED	3	
		ABS,X	FD	3	
		ABS,Y	F9	3	
		(Z,X)	E1	2	
		(Z),Y	F1	2	
SEC	置进位	IMP	38	1	C＝1
SED	置十进制位	IMP	F8	1	D＝1
SEI	置中断屏蔽位	IMP	78	1	I＝1
STA	$A\rightarrow M$	Z	85	2	
		Z,X	95	2	
		ABS	8D	3	
		ABS,X	9D	3	
		ABS,Y	99	3	
		(Z,X)	81	2	
		(Z),Y	91	2	
STX	$X\rightarrow M$	Z	86	2	
		Z,X	96	2	
		ABS	8E	3	
STY	$Y\rightarrow M$	Z	84	2	
		Z,X	94	2	
		ABS	8C	3	

（续上表）

记忆符	意　　义	寻址方法	操作码	字节	影响标志
TAX	A→X	IMP	AA	1	NZ
TAY	A→Y	IMP	A8	1	NZ
TSX	栈指示器S→X	IMP	BA	1	NZ
TXA	X→A	IMP	8A	1	NZ
TXS	X→栈指示器S	IMP	9A	1	
TYA	Y→A	IMP	98	1	NZ

注：在上表中，影响标志除表明该标志等于0之外，其余写出的标志被置“1”。

6502的指令有单字节、双字节、三字节三种格式，如下图一所示。



图一　指令格式

一、寻址方法

6502的13种寻址方法是：绝对寻址、累加器寻址、立即寻址、隐含寻址、绝对X变址寻址、绝对Y变址寻址、零页寻址、零页X变址寻址、零页Y变址寻址、相对寻址、间接寻址、先X变址后间接寻址、先间接后Y变址寻址。

1．绝对寻址（简记：ABC）

这种寻址方法的指令是三字节指令。其第二、三字节给出的就是操作数地址。一般有：

有效地址＝（指令第二、三字节）

用括号（　）括起来是表示第二、三字节的内容。绝对寻址有时也称直接寻址。

要注意，第三字节的内容是高位地址。例如：ADC $2160

2．累加器寻址（简记：A）

这类指令是单字节指令。只对累加器A的内容执行操作的指令，称为累加器寻址指令。比如A左移一位指令：ASL A

3．立即寻址（简记：IMM引导符号用“#”）

这类指令是双字节指令。直接用本指令中的第二个字节的内容作为数据执行操作的指令，叫做立即寻址指令。例如：LDA #$7D

4．隐含寻址

这类指令是单字节指令。无需操作数地址的指令，称作隐含寻址指令。这类指令实质上并无寻址方法。例如，中止指令BRK；变址器X加1指令INX等。

5．绝对X变址寻址（简记：ABS，X）

这类指令是三字节指令，其中第二、三字节是绝对地

址。在这类指令中，有效地址是绝对地址加上X变址寄存器的内容：

有效地址＝绝对地址＋（X）

6．绝对Y变址寻址（简记：ABS，Y）

这类指令是三字节指令，其中第二、三字节是绝对地址。在这类指令中，有效地址是绝对地址加上Y变址寄存器的内容：

有效地址＝绝对地址＋（Y）

7．零页寻址（简记：Z）

这类指令是双字节指令，其中第二字是低位地址，高位地址为00。由于6502的高位地址（8位）用作存贮器页面地址，低位地址（8位）用作页面存贮单元地址，故此6502的存贮器页面有00～255共256个，一个页面有存贮单元00～255共256个。在零页寻址指令中，只有二个字节。第二个字节用于表示页面存贮单元地址；无第三字节即无页面地址，也就是页面地址为00，这也是零页寻址的来源。在这种寻址方法中，有效地址是0000加第二字节内容，因而第二字节也称为零页地址。即：

有效地址＝0000＋（第二字节）

8．零页X变址寻址（简记：Z，X）

这类指令是双字节指令，其中第二字节是低位地址，也是零页地址。有效地址是零页地址加X变址寄存器的内容。即：

有效地址＝0000＋（第二字节）＋（X）

例如：LDA　$3B，X

9．零页Y变址寻址（简记：Z，Y）

这类指令是双字节指令，其中第二字节是低位地址，也

是零页地址。有效地址是零页地址加Y变址寄存器的内容。即：

有效地址 = 0000 + （第二字节）+（Y）例如：LDX $27，Y

10．相对寻址（简记：REL）

这类指令是双字节指令。有效地址是程序计数器加第二字节的内容，即：

有效地址 =（PC）+（第二字节）

相对寻址一般用在转移指令。指令的第二字节实际是相对转移量，转移量的范围为 -127～+127。当转移量为正时，用原码表示；转移量为负时用补码表示。如：

+35——00100011 = $23

-35——11011101 = $DD

11．间接寻址（简记：（ABS））

间接寻址指令只有一条，就是JMP指令。JMP是三字节指令。指令的第二、三字节是操作数地址的地址。也就是说，把用第二、三字节作地址去找到的内容作为地址，再去取出的内容才是真正操作数，即：

有效地址 =（（第二，三字节））

12．先X变址后间接（简记：（Z，X））

这类指令是两字节指令。X变址就是：基本（零页）地址的内容加X的内容得到间接地址。间接就是由间接地址的内容产生有效地址的低位。该间接地址的下一个地址的内容产生有效地址的高位。指令就从有效地址中取得操作数。即：

先X变址：（（Z）+ X）= 间接地址

后间接：（间接地址）= 有效地址的低位L

（间接地址＋1）＝有效地址的高位H

有效地址＝HL

13．先间接后Y变址（简记：(Z)；Y）

这类指令是两字节指令。第二字节是零页地址，用作间接寻址的间接地址。

先间接：由间接地址的内容产生基本地址的低位L'，该间接地址的下一个地址的内容产生基本地址的高位H'。

后Y变址：由上所得到的基本地址H'L'加上Y的内容得到有效地址。

指令便从有效地址中取得操作数。即：

先间接：(Z)＝L'

(Z＋1)＝H'

H'L'＝基本地址

后Y变址：H'L'＋Y＝有效地址。

在上述所有寻址方法中，用得比较多的是绝对寻址、累加器寻址、立即寻址、隐含寻址、零页寻址、相对寻址、零页X变址寻址等，零页Y变址寻址和间接寻址用得较少，它们各自只用在一条指令中。零页Y变址寻址只有在变址器X送存贮器指令STX时才用到，间接寻址只有在无条件转移指令JMP时用到。寻址方法的简写如表二所示。

表二 寻址方法及简写

英文	寻址方法说明	简写
Acumulator	累加器	A
Absolute	绝对	ABS
Absolute, X	绝对X变址	ABS,X
Absolute, Y	绝对Y变址	ABS,Y
Immediate	立即	IMM
Implied	隐含	IMP
Indirect	间接	(ABS)
Indirect, X	先X变址,后间接	(Z, X)
Indirect, Y	先间接,后Y变址	(Z), Y
Relative	相对	REL
Zero Page	零页	Z
Zero Page, X	零页X变址	Z, X
Zero Page, Y	零页Y变址	Z, Y

二、指令系统及其功能

6502指令系统按指令功能分，基本可分成9类，下面分别加以介绍。

1. 算术逻辑指令

6502的算术逻辑指令有ADC、AND、EOR、ORA、SBC五条。它们分别执行带进位加、“与”、“或”、“异或”、带借位减操作。运算结果存放在累加器A中。

2. 移位指令

移位指令有ASL、LSR、ROL、ROR四条。它们分别表示左移一位，右移一位，循环左移一位，循环右移一位。这类指令有五种寻址方法：累加器寻址，零页寻址，零页X变址寻址，绝对寻址，绝对X变址寻址。在累加器寻址方式

中，指令对累加器A中的内容执行操作；在其余寻址方法中，指令对存贮器中的内容执行操作。

3. 置标志指令

这类指令有清标志指令CLC、CLD、CLI、CLV四条，置标志指令SEC、SED、SEI三条。它们分别对状态寄存器P中的标志C、D、I、V清零，以及对C、D、I置"1"。

4. 比较指令

比较指令有BIT、CMP、CPX、CPY。其中BIT是把A和M进"与"操作，以取得D_7和D_6的结果，再送入N、V中。其余指令则是分别把A、X、Y的内容减去M的内容，以其结果来置标志位N、Z、C。比较指令和逻辑指令不同，比较指令的比较结果数据不再放回A中。

5. 转移指令

这类指令包括条件转移指令BCC、BCS、BEQ、BMI、BNE、BPL、BVC、BVS；无条件转移指令JMP、JSR；返回指令RTI、RTS。

条件转移指令视条件是否满足而决定去向。满足时转到由指令第二字节所指明的地址，不满足则不转；如BCC是非进位转移指令，即进位标志C＝0则转移，否则不转移。无条件转移指令则无条件转到指令第二、三字节所指明的地址中。返回指令回到返回断点。

6. 送数指令

送数指令有LDA、LDX、LDY、STA、STX、STY、TAX、TAY、TSX、TXA、TXS、TYA共十二条。这些指令执行的操作是把寄存器的内容送入内存，或是把内存的内容送到寄存器中，或是把一个寄存器的内容送入另一个寄存器中。例如：LDA指令是把存贮器的内容送到A中；STA

指令是把A的内容送入存贮器；TXA指令是把变址寄存器X的内容送到A中。注意：把数据从一个地址送到另一个地址是没有直接的指令的，要通过A或X或Y才能实现两个地址之间的数据传送。

7．增量指令

这类指令包括加1指令INC、INX、INY以及减1指令DEC、DEX、DEY。它们分别执行对存贮器X、Y加1，或者对存贮器X、Y减1。例如，DEX指令执行把变址寄存器X的内容减1的操作。

8．堆栈指令

堆栈指令包括入栈指令PHA、PHP和出栈指令PLA、PLP。堆栈指令的功能不多，它们只对累加器A或状态寄存器P执行操作。例如：PHA指令执行把A入栈的操作；PLP执行使P出栈的操作。

9．停机和空操作指令

停机指令是BRK，它强迫微处理机中止执行。空操作指令是NOP，它不做任何操作。

6502上述9种指令类型的基本指令有56条。由于有的指令可以有多种寻址方式，故此，6502的指令在实际使用中可以扩展为151条具体指令。

三、6502指令助记符英文全称

ADC	Add Memory to Accumulator with Carry
AND	"AND" Memory with Accumulator
ASL	Shift Left One Bit (Memory or Accumulator)
BCC	Branch on Carry Clear
BCS	Branch on Carry Set

BEQ	Branch on Result Zero
BIT	Test bits in Memory with Accumulator
BMI	Branch on Result Minus
BNE	Branch on Result not Zero
BPL	Branch on Result Plus
BRK	Force Break
BVC	Branch on Overflow Clear
BVS	Branch on Overflow Set
CLC	Clear Carry Flag
CLD	Clear Decimal Mode
CLI	Clear interrupt disable bit
CLV	Clear overflow flag
CMP	Compare memory and accumulator
CPX	Compare Memory and Index X
CPY	Compare Memory and index Y
DEC	Decrement memory by one
DEX	Decrement index X by one
DEY	Decrement index Y by one
EOR	"Exclusive—Or" memory with accumulator
INC	Increment memory by one
INX	Increment Index X by one
INY	Increment Index Y by one
JMP	Jump to New Location
JSR	Jump to New Location Saving Return Address

LDA	Load Accumulator with Memory
LDX	Load Index X with Memory
LDY	Load Index Y with Memory
LSR	Shift Right one bit (Memory or Accumulator)
NOP	NO Operation
ORA	"OR" Memory with Accumulator
PHA	Push Accumulator on stack
PHP	Push Processor Status on Stack
PLA	Pull Accumulator from stack
PLP	Pull Processor Status from Stack
ROL	Rotate One Bit Left (memory or accumulator)
ROR	Rotate One Bit Right (memory or accumulator)
RTI	Return from Interrupt
RTS	Return from Subroutine
SBC	Subtract Memory from Accumulator with Borrow
SEC	Set carry flag
SED	Set decimal mode flag
SEI	Set interrupt disable status
STY	Store index Y in Memory
STA	Store accumulator in Memory
STX	Store index X in Memory
TAX	Transfer Accumulator to Index X
TAY	Transfer Accumulator to Index Y

TYA	Transfer Index Y to Accumulator
TSX	Transfer stack pointer to index X
TXA	Transfer Index X to Accumulator
TXS	Transfer Index X to Stack Pointer

附录五　APPLE Ⅱ机内存空间的分配

APPLE Ⅱ微型计算机内存空间的256页存贮区，一共分成三类：（1）随机存取存贮单元（RAM）；（2）只读存贮单元（ROM）；（3）输入输出（I/O）位置。不同的存贮区有不同的功能。表一为APPLE Ⅱ机的内存贮空间分配表（见311页）。

从存贮空间分配表中可知，RAM存贮区是从第0页开始，一直到191页结束。在主电路板上，RAM的内存容量可由16K到48K（49152字节），此外用户也可以另加一块16K的RAM板，使得RAM由48K扩充至64K。

大部分的RAM是用来存用户的程序和数据，但也保留了一些区域给系统监控程序、计算机语言及其它系统函数使用。

下面表二列出RAM存贮区的结构和用途（见312页）。

表一　　系统存贮空间分配表

页数		功能
十进制	十六进制	
0	$00	
1	$01	
2	$02	
⋮	⋮	RAM (48K)
190	$BE	
191	$BF	
192	$C0	
193	$C1	
⋮	⋮	I/O (2K)
198	$C6	
199	$C7	
200	$C8	
201	$C9	
⋮	⋮	I/O ROM (2K)
206	$CE	
207	$CF	
208	$D0	
209	$D1	
⋮	⋮	ROM (12K)
254	$FE	
255	$FF	

表二　　RAM结构和用途

<table>
<tr><th>页
十进制</th><th>数
十六进制</th><th colspan="2">用　　于</th></tr>
<tr><td>0</td><td>$00</td><td colspan="2">系统程序</td></tr>
<tr><td>1</td><td>$01</td><td colspan="2">系统堆栈</td></tr>
<tr><td>2</td><td>$02</td><td colspan="2">GETLN输入缓冲区</td></tr>
<tr><td>3</td><td>$03</td><td colspan="2">监控程序向量地址</td></tr>
<tr><td>4
5
6
7</td><td>$04
$05
$06
$07</td><td colspan="2">文本和低分辨率图形首页存贮区</td></tr>
<tr><td>8
9
10
11</td><td>$08
$09
$0A
$0B</td><td>文本和低分辨率图形次页存贮区</td><td rowspan="5">用户可使用的RAM</td></tr>
<tr><td>12
至
31</td><td>$0C
至
$1F</td><td></td></tr>
<tr><td>32
至
63</td><td>$20
至
$3F</td><td>高分辨率图形首页存贮区</td></tr>
<tr><td>64
至
95</td><td>$40
至
$5F</td><td>高分辨率图形次页存贮区</td></tr>
<tr><td>96
至
191</td><td>$60
至
$BF</td><td></td></tr>
</table>

下面解释几个存贮区的特殊用途。

零页（ZERO PAGE）：零页是机器语言程序使用的重要场所。系统监控程序用了这里将近20个地址，INT BASIC用更多一点，而FP BASIC和磁盘操作系统，用了剩下的第零页存贮区。详见表三～表六。

第一页：6502微处理器将第一页的256个字节，全部保留给堆栈("Stack")使用。尽管通常都不会用去这一页的一半位置，但用户也无法知道到底用了多少，所以最好不要在第一页存任何数据。

第二页：此页保留给GETLN子程序，作为其在输入一行字符时的输入缓冲区。

第三页：监控程序使用了第三页的最上面十六个地址，即＄3F0到＄3FF（十进制的地址1008到1023）。

第四到第七页：这四页共1024个字节的存贮区是用来做文本型和低分辨率图形型的显示暂存区。因此这些地址是不能用来存放其它数据的。但这里共有64个地址并没有显示在屏幕上，这些地址是保留给外围扩充板使用的。

表三

零页存贮区图

（监控使用的零页区）

十进		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	十六进	$0	$1	$2	$3	$4	$5	$6	$7	$8	$9	$A	$B	$C	$D	$E	$F
0	$00																
16	$10																
32	$20	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
48	$30	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
64	$40	•	•	•	•	•	•	•	•	•	•					•	•
80	$50	•	•	•	•	•	•										
96	$60																
112	$70																
188	$80																
144	$90																
160	$A0																
176	$B0																
192	$C0																
208	$D0																
224	$E0																
240	$F0																

注：小圆黑点所在的地址表示已被使用，空白则未用（以下几个图均同）。

表四　　　　FP BASIC 使用的零页区

十进	十六进	0 $0	1 $1	2 $2	3 $3	4 $4	5 $5	6 $6	7 $7	8 $8	9 $9	10 $A	11 $B	12 $C	13 $D	14 $E	15 $F
0	$00	•	•	•	•	•	•					•	•	•	•	•	•
16	$10	•	•	•	•	•	•	•	•	•							
32	$20																
48	$30																
64	$40																
80	$50	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
96	$60	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
112	$70	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
128	$80	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
144	$90	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
160	$A0	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
176	$B0	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
192	$C0	•	•	•	•	•	•	•	•	•	•	•	•	•	•		
208	$D0	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
224	$E0	•	•	•		•	•	•	•	•	•	•					
240	$F0	•	•	•	•	•	•	•	•	•							

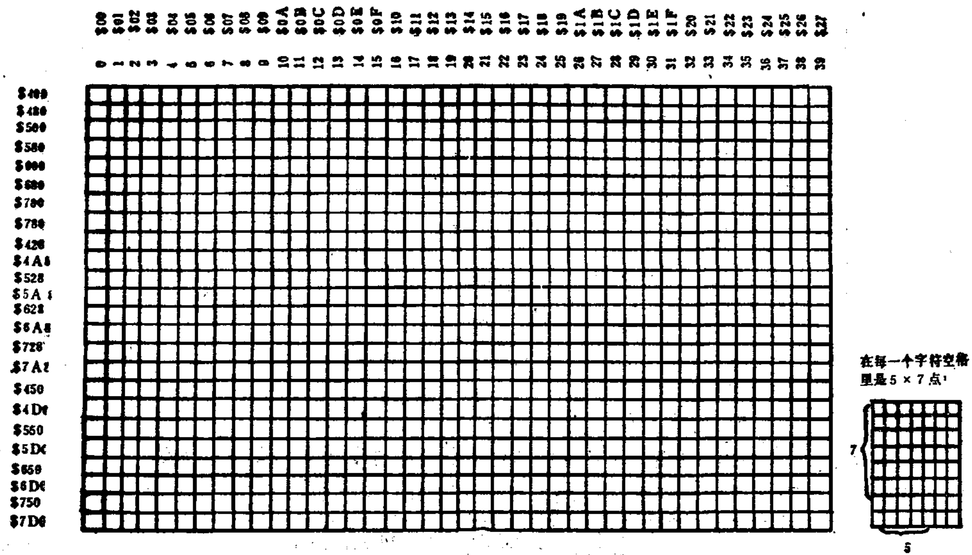
表五 DOS3.3 使用的零页区

十进		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	十六进	$0	$1	$2	$3	$4	$5	$6	$7	$8	$9	$A	$B	$C	$D	$E	$F
0	$00																
16	$10																
32	$20					•		•	•			•	•	•	•	•	•
48	$30				•		•	•	•	•	•			•	•	•	•
64	$40	•	•	•	•	•	•	•	•	•	•	•	•	•	•		
80	$50																
96	$60								•	•	•	•					•
112	$70	•			•	•		•									
128	$80																
144	$90																
160	$A0																•
176	$B0	•															
192	$C0											•	•	•	•		
208	$D0							•		•	•						
224	$E0																
240	$F0																

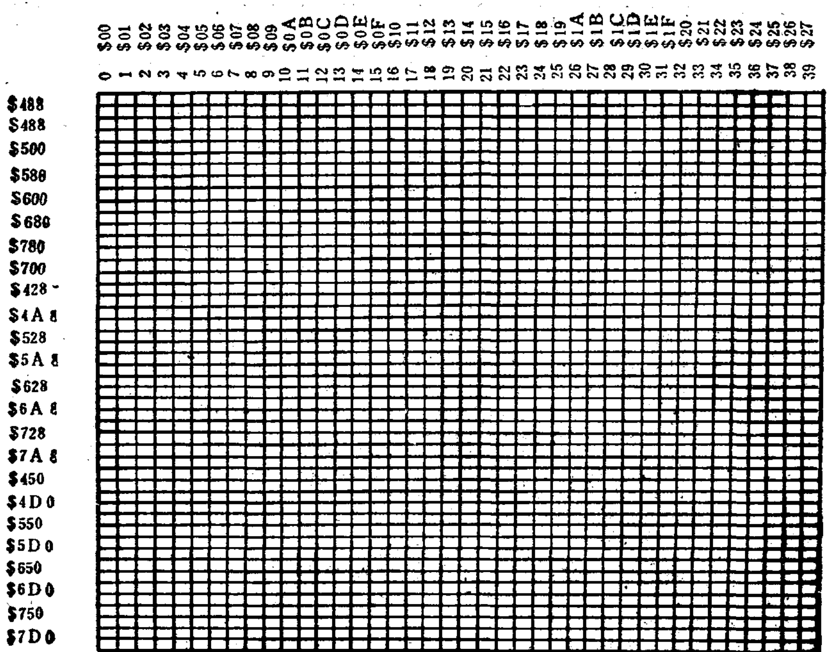
表六 INT BASIC 使用的零页区

十进		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	十六进	$0	$1	$2	$3	$4	$5	$6	$7	$8	$9	$A	$B	$C	$D	$E	$F
0	$00																
16	$10																
32	$20																
48	$30																
64	$40											•	•	•	•		
80	$50						•	•	•	•	•	•	•	•	•	•	•
96	$60	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
112	$70	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
128	$80	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
144	$90	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
160	$A0	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
176	$B0	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
192	$C0	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
208	$D0	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
224	$E0																
240	$F0																

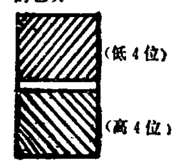
附录六　正文显示屏与显示RAM的关系图



附录七　低分辨率图形屏幕与显示RAM的关系图

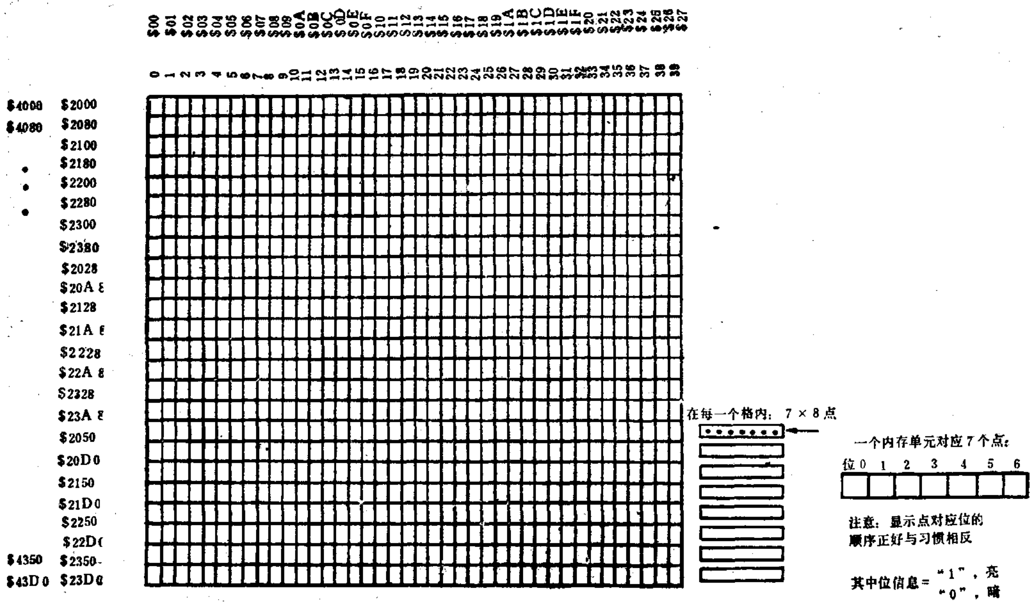


在每个方格格内
是两个上下相叠
的色块



低 4 位和高 4 位的值
将分别确定着上下
两个小色块的颜色。

附录八　高分辨率图形屏幕与显示RAM 的关系图



附录九 ASCII 屏幕字符表

十进	十六进	反相显示 0 $00	16 $10	32 $20	48 $30	闪烁显示 64 $40	80 $50	96 $60	112 $70	控制 128 $80	144 $90	正常 160 $A0	176 $B0	192 $C0	208 $D0	小写体 224 $E0	240 $F0
0	$0	@	P		0	@	P		0	@	P		0	@	P		0
1	$1	A	Q	!	1	A	Q	!	1	A	Q	!	1	A	Q	!	1
2	$2	B	R	"	2	B	R	"	2	B	R	"	2	B	R	"	2
3	$3	C	S	#	3	C	S	#	3	C	S	#	3	C	S	#	3
4	$4	D	T	$	4	D	T	$	4	D	T	$	4	D	T	$	4
5	$5	E	U	%	5	E	U	%	5	E	U	%	5	E	U	%	5
6	$6	F	V	&	6	F	V	&	6	F	V	&	6	F	V	&	6
7	$7	G	W	'	7	G	W	'	7	G	W	'	7	G	W	'	7
8	$8	H	X	(	8	H	X	(	8	H	X	(	8	H	X	(	8
9	$9	I	Y	)	9	I	Y	)	9	I	Y	)	9	I	Y	)	9
10	$A	J	Z	*	:	J	Z	*	:	J	Z	*	:	J	Z	*	:
11	$B	K	[	+	;	K	[	+	;	K	[	+	;	K	[	+	;
12	$C	L	\	,	<	L	\	,	<	L	\	,	<	L	\	,	<
13	$D	M	]	—	=	M	]	—	=	M	]	—	=	M	]	—	=
14	$E	N	∧	.	>	N	∧	.	>	N	∧	.	>	N	∧	.	>
15	$F	O	—	/	?	O	—	/	?	O	—	/	?	O	—	/	?

附录十　十六进制数与十进制数转换表

HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	00	000
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	0
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	256	4096
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	512	8192
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	768	12288
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	1024	16384
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	1280	20480
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	1536	24576
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	1792	28672
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	2048	32768
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	2304	36864
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	2560	40960
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	2816	45056
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	3072	49152
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	3328	53248
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	3584	57344
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	3840	61440

附录十一　十进制数与 BCD 转换表

十进数	BCD	DEC	BCD
0	0000	16	00010110
1	0001	17	00010111
2	0010	18	00011000
3	0011	19	00011001
4	0100	90	10010000
5	0101	91	10010001
6	0110	92	10010010
7	0111	93	10010011
8	1000	94	10010100
9	1001	95	10010101
10	00010000	96	10010110
11	00010001	97	10010111
12	00010010	98	10011000
13	00010011	99	10011001
14	00010100		
15	00010101		

5		4		3		2		1		0	
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0	0	0	0	0
1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1
2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2
3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3
4	4,194,304	4	262,144	4	16,384	4	1,024	4	64	4	4
5	5,242,880	5	327,680	5	20,480	5	1,280	5	80	5	5
6	6,291,456	6	393,216	6	24,576	6	1,536	6	96	6	6
7	7,340,032	7	458,752	7	28,672	7	1,792	7	112	7	7
8	8,388,608	8	524,288	8	32,768	8	2,048	8	128	8	8
9	9,437,184	9	589,824	9	36,864	9	2,304	9	144	9	9
A	10,485,760	A	655,360	A	40,960	A	2,560	A	160	A	10
B	11,534,336	B	720,896	B	45,056	B	2,816	B	176	B	11
C	12,582,912	C	786,432	C	49,152	C	3,072	C	192	C	12
D	13,631,488	D	851,968	D	53,248	D	3,328	D	208	D	13
E	14,680,064	E	917,504	E	57,344	E	3,584	E	224	E	14
F	15.728,640	F	983,040	F	61,440	F	3,840	F	240	F	15

附录十二　相对转移位移量查算表

向前相对转移（正转移）

MSD（高位） \ LSD（低位）	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127

果粉藏书

向后相对转移（负转移）

MSD（高位） \ LSD（低位）	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	128	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113
9	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97
A	96	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81
B	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65
C	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49
D	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33
E	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17
F	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

中国科技图书公司发票

地址：上海河南中路221号 电话：234·67

开户行黄办南分处 发票帐号 3565015

监制章 9

户名

类别	册数	金额
科学技术图书	1	1.60
一般图书		
合计	1	1.60
人民币（大写） 壹元陆角		

198 9 年 1 月 9 日 经手人

上海市工商企业登记印刷所 黄字...

收讫

A. № 0527996

数字涂改 未盖收款章者无效

微型电子计算机教程

书名	定价
微型计算机基础	定价：1.60元
TRS—80微型计算机 BASICⅡ语言与磁盘操作系统	定价：1.60元
APPLEⅡ 微型计算机 BASIC语言与磁盘操作系统	定价：1.60元
APPLEⅡ 微型计算机 系统结构	定价：1.60元
APPLEⅡ 微型计算机 实习操作指导	定价：1.60元
R1微型计算机读本	定价：1.50元

统一书号 15182·119
定 价 1.60 元